# Single and dual energy computed tomography in proton therapy

by

Sandra M. S. Reitan

Thesis

for the degree of

Master in Biophysics and Medical Physics


Faculty of Mathematics and Natural Sciences

University of Oslo

Spring 2021

# Acknowledgment

First, I would like to thank my supervisors; Taran P. Hellebust, Eirik Malinen, and Kristin Jensen – a sincere thank you for the guidance, discussions, and motivation throughout this project. To the Biophysics and Medical Physics group at the Department of Physics at the University of Oslo, thank you for this opportunity and great conversations over lunch.

A special thanks to my family, Audun S. Lien, Bjørn H. Reitan, Turid Storvik and Linn Reitan; thank you for the patience and support you have shown me during this project. Last but not least, to all of my extremely supporting friends; I truly appreciate you.

Sandra M. S. Reitan

June 2021

# Abstract

**Purpose:** The range uncertainties in proton therapy is a problem in current treatment planning of cancer, and is largely due to limitations in estimating stopping power (S) from computed tomography (CT) images. With a dual energy CT (DECT) scanning, monochromatic images can be provided that can potentially reduce noise and lead to better S estimates. In the current work, DECT is employed to investigate the impact of varying x-ray energy in conversion from Hounsfield Unit (HU) to S and proton treatment plan quality.

**Methods and materials:** We reconstructed monochromatic images from 40 keV to 140 keV at 10 keV increments from data acquired by DECT and a 120 kVp CT image for two phantoms (Gammex and Quasar). We estimated Hounsfield Units (HU) values of different media (HE blood40, HE blood70, HE brain, CT HE solid water, HE blood100, HE adipose, lung, dense bone, water equivalent, inner bone, and polyethylene) and calculated the contrast to noise ratio (CNR) in each monochromatic image. From these calculations we found the monochromatic image with HU values and noise closest to the 120 kVp single energy CT (SECT) image. We then simulated a CT scanner in python and found the HU value for different media (Soft tissue, Blood, Brain, Inflated lung, Cortical bone, Adipose, and polyethylene) for both a monochromatic x-ray beam and 120 kVp x-ray beam. Stopping power ratio (SPR) was calculated for corresponding media, which was used to see investigate how conversion from HU to SPR changes with energies and noise in an image. DECT and CT was also done for two anthropomorphic head phantoms. The HU values for three media (soft tissue, bone, air) were collected in four different locations inside the head. These values were used to find a standard deviation between the locations. The 120 kVp image was then used to make a proton treatment plan for two different targets. The dose plans were recalculated in the monochromatic images to see how this would affect the dose calculation. A new HU-SPR CT calibration from the two first phantoms were used to see how the dose plan changes with CT calibration.

**Results:** Measured and simulated Hus corresponded quite well across the different phantoms and material inserts. SPR was shown to depend roughly linearly with HU, but the relation changed with x-ray energy. In the DECT images a change in HU with energy and material compositions was seen a best fit to the 120 kVp SECT image. Comparing the HU-SPR relations for 40 keV, 80 keV, 140 keV, and 120 kVp showed a straighter line for the 80 keV DECT image. Proton planning with the SECT image series and different DECT images gave some variations in proton plan quality.

**Conclusion:** Monochromatic images from DECT gave variations in Hus, leading to differences in HU-SPR conversion. If not accounted for, this may result in deterioration in proton plan quality. Still, DECT may improve proton plan quality if appropriate procedures are undertaken.

# Abbreviations

| Abbreviation | Description |
|---|---|
| BP | Backprojection |
| CNR | Contrast to noise ratio |
| CT | Computed tomography |
| $CTDI_{vol}$ | Computed Tomography Dose Index Volume |
| CSDA | Continuously Slowing Down Approximation |
| CTV | Clinical target volume |
| DECT | Dual Energy CT |
| DNA | Deoxyribonucleic acid |
| FOV | Field of View |
| GTV | Gross target volume |
| HLUT | Heuristic look up table |
| HU | Hounsfield Unit |
| IMPT | Intensity modulated proton treatment |
| IT | Iterative reconstruction |
| MFO | Multi field optimisation |
| NIST | National Institute of Standards and Technology |
| PBS | Pencil beam scanning |
| PTV | Planning target volume |
| OAR | Organs at risk |
| RT | Radiotherapy |
| RBE | Relative biological effectiveness |
| SECT | Single energy CT |
| SFOV | Scan field of view |
| SFUD | Single field uniform dose |
| S | Stopping power |
| SOBP | Spread out Bragg peak |
| SPR | Stopping power ratio |

# *Table of content*

# 1 Introduction

Each year about 20 000 people get diagnosed with cancer in Norway. Surgery, chemotherapy, and/or radiation therapy are the three treatments available, whereas about 7000 of the patients are treated with radiation therapy as the main treatment or in combination with one of the other treatments [1].

Cancer cells are cells which proliferate uncontrolled. This genetic error makes it important to eradicate all cancer cells during treatment. Radiation therapy, also known as radiotherapy, is an important modality for cancer treatment. It attacks the tumour with ionising radiation to induce cell death by damaging the *deoxyribonucleic acid* (DNA) within the cancer cells [2]. The main aim of radiotherapy is to give a large dose of radiation to the tumour while sparing the normal tissue. Normal tissue includes *organs at risk* (OAR) and other healthy tissue.

In radiotherapy (RT), *Computed tomography* (CT) images are used to image the anatomy of the patient in treatment position. With the CT images in treatment planning, it is possible to delineate the tumour and OAR as well as to conform the radiation beams so the tumour is irradiated while maintaining a low radiation dose to the normal tissue. The CT images are also used in calculations of the radiation dose to the patient by providing information about the material composition [3].

Today the radiotherapy in Norway consists of x-ray radiation (external radiation), brachytherapy (internal radiation), and electron therapy. Proton therapy is an upcoming treatment technique that may be preferably for many cancer sites due to highly localized energy deposition at the proton track end – “Bragg peak”. The coming proton treatment centre in Oslo will have a treatment capacity about 850 patients annually when the treatment rooms are in operation [4]. To be able to take full advantage of the benefits proton therapy gives, current uncertainties in proton range prediction from CT must be further minimized [5].

The uncertainties in proton therapy are not the same as for photon-based RT. Specifically, this relates to the use of CT information to calculate the transport of protons in tissue. Key interest is the estimation of *Stopping power ratio* (SPR), which is a key factor in determining tissue radiation dose from charged particles. We have seen developments in CT technology leading to the concept of *Dual Energy CT* (DECT). With this new technology it could be possible to get a more accurate tissue characterization with might lead to a more precise conversion of *Hounsfield Unit* (HU; the radiodensity in a given voxel in a CT image) to SPR. The hypothesis is that the use of DECT for proton treatment planning would reduce some of the uncertainties in proton therapy.

Today a heuristic conversion of HU into SPR is done by a universal heuristic look-up table (HLUT), an additional margin of about 3.5% of absolute range are used clinically to account for these uncertainties [5]. Studies have found a higher accuracy for DECT than CT for SPR

estimation [6]. DECT acquires two CT acquisitions with different energy (80/140kV) which gives the benefit of differentiating between energy-dependent image contrasts. The attenuation of x-ray through the material for the two energies will be different, and from analysing differences in attenuation it is possible to find the material composition. This information can be used to calculate monochromatic images. A monochromatic image is reconstructed CT images derived from DECT with different photon energies (keV).

The main goal with DECT in proton therapy is to obtain a direct SPR calibration to minimize the range uncertainties, and by this achieve a more precise dose calculation. As a step on the way, clinical implementation of DECT and monochromatic images as the main image type for treatment planning is an important part. In this work, we investigated change in HU values for monochromatic CT images derived from DECT and 120 kVp CT images, and how the images impact the conversion from HU to SPR. Eleven reconstructed monochromatic DECT images were investigated. We performed treatment planning using the 120 kVp image basis and investigated how the plan changed when changing the CT image basis and HU-SPR conversion formulae.

# 2 Theory

## 2.1 Charged particle interactions

Charged particle interactions are important in proton therapy and will be explained in the following chapter, *which is based on Chapter 8 of Radiological physics and Radiation Dosimetry by Attix (2008)* [7].

This chapter will focus on the two charged particles; protons and electrons. Protons are positively charged particles while electrons are negatively charged particles, in addition the proton has a mass almost twice the size of the electron. Charged particles interact by the *Coulomb force,* and since the accelerated charged particles lose its energy almost continuously through interactions in the matter, it is possible to roughly characterise a pathlength.

An energetic charged particle can interact in different ways depending on the distance between the particle and the atom it will interact with. In figure 2.1 a proton passing an atom with a distance $b$ (i.e., the impact parameter), and the radius of the atom $a$ is shown. The three dominant interactions are; if the particle passes the atom far outside the radius of the atom ($b \gg a$), if the particle path is close to the atom radius ($b \sim a$), and if the particle passes close to the nucleus ($b \ll a$), they are called *soft collision*, *hard collision,* and *Coulomb force interactions with the external nuclear field*, respectively.

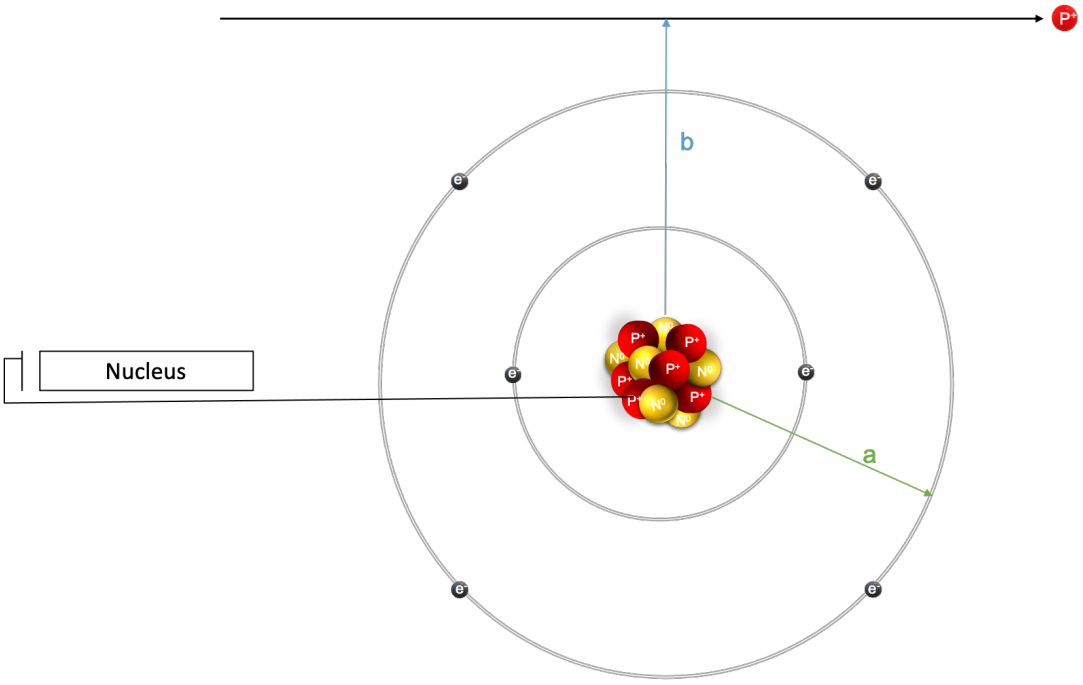


*Figure 2.1: Important parameters in charged particle collisions, a; is the classical atomic radius, b is the classical impact parameter. The atomic nucleus is the made of protons (in red) and neutrons (in yellow). [7]*

### 2.1.1 Soft Collisions ($b \gg a$)

In the case of a soft collision, the particle passes the atom with a considerable distance compared to the radius, as seen in figure 2.1. The Coulomb force between the particle and atomic electrons causes an interaction. Consequentially, the whole atom will be affected. In a soft collision the particle will transfer a very small amount of its energy to the atom, with the likely outcome for the atom is a distortion, excitation, or ionization of an electron. Soft collision is an interaction that happens most frequently in matter, and which will stand for roughly half of the transferred energy to the absorbing medium.

### 2.1.2 Hard ("knock – on") Collisions ($b \sim a$)

When *b* in figure 2.1 is approximately the same as the radius of the atom, it is more likely that the particle will interact with a single atomic electron instead of the whole atom, this is called a hard collision. The single electron will be ionized from the atom and it will have a considerable amount of kinetic energy. This electron is commonly called "delta-ray" and is able to move a substantial distance and will have enough energy to experience Coulomb interaction on its own. The path of this delta-ray is called a "spur". If the ejected electron originally was in the inner shell of the atom, characteristic x-ray (and/or Auger electrons) will also be emitted with the electron. Hard collision is not as common as the soft collision, but the total energy amount transferred to the medium will be approximately the same for the two collisions. This is due to the higher amount of transferred energy in a hard collision vs. a soft collision.

### 2.1.3 Coulomb force interactions with the external nuclear field ($b \ll a$)

The probability of interaction between the proton and the nucleus is *very small*, and for this inelastic collision to take place the particle energy must be sufficiently high ($\sim 100 MeV$). When the particle hits one or more nucleons (protons or neutrons), the latter might be ejected from the nucleus. The target nucleus may decay from its excited state by emission of particles and $\gamma$-rays.

### 2.1.4 Stopping Power

*Stopping power (S)* describes the energy loss for charged particles as they travers a medium. It is defined as the energy loss (of the charged particle) per unit length, see equation 2.1.

$$S = -\frac{dE}{dx} \qquad \text{Eq. 2.1}$$

The typical unit for the stopping power is $\frac{MeV}{cm}$. In *Attix* [7] the mass stopping power is given by the Bethe-Bloch formula:

$$\frac{dE}{\rho dx} = 4\pi r_0^2 m_e c^2 \left(\frac{N_A Z}{A}\right)\left(\frac{z}{\beta}\right)^2 \left[ln\left(\frac{2m_e c^2 \beta^2}{(1-\beta^2)I}\right) - \beta^2\right] \qquad \text{Eq. 2.2}$$

were $\beta = \left[1 - \left(\frac{1}{\left(T/_{M_0 c^2}\right)+1}\right)^2\right]^{\frac{1}{2}}$.

*Mass* stopping power has unit $\frac{MeV \cdot cm^2}{g}$, which thus is independent of the density of the absorbing medium. Mass stopping power equation includes both soft and hard collisions. In table 2.1 the definition of the different variables used in equation 2.2 are listed.

*Table 2.1:Parameters and definitions in the Bethe-Bloch formula.*

| ***Symbol*** | ***Definition*** |
|---|---|
| $\rho$ | Mass density of the absorbing medium |
| $N_A Z/_A$ | Number of electrons per gram of the stopping medium |
| I | Mean excitation energy of the absorbing medium |
| z | Charge of the particle (proton) |
| $m_e c^2$ | Rest energy of the electron [0.511MeV] |
| $4\pi r_0^2$ | A constant |
| T | Kinetic energy of the proton |
| $M_0c^2$ | Rest energy of the proton [938.28 MeV] |

For the Stopping power we then get:

$$\frac{dE}{dx} = 4\pi r_0^2 m_e c^2 \left(\frac{N_A Z}{A} \cdot \rho\right)\left(\frac{z}{\beta}\right)^2 \left[ln\left(\frac{2m_e c^2 \beta^2}{(1-\beta^2)I}\right) - \beta^2\right]$$

$$\frac{dE}{dx} = \rho_e \cdot \ 4\pi r_0^2 m_e c^2 \left(\frac{z}{\beta}\right)^2 \left[ln\left(\frac{2m_e c^2 \beta^2}{(1-\beta^2)I}\right) - \beta^2\right] \qquad \text{Eq 2.3}$$

were $\rho_e = (\frac{N_A Z}{A} \cdot \rho)$ is the electron density, i.e., the number of electrons per unit volume. Schneider's [8] key article was used for understating how the stopping power was dependent on the electron density. But the article erroneously reported $\frac{N_A Z}{A}$ as the electron density per volume.

To calculate the pathlength or *range* of the proton, stopping power is used. Range straggling affects the pathlengths of the protons and is a result of stochastic variations in rates of energy loss.

### 2.1.5 Range

Definitions (p. 180 Attix) of the concept charged particle *range:*

> *The range ℛ of a charged particle of a given type and energy in a given medium is the expectation value of the pathlength $p$ that it follows until it comes to rest.*

The *Continuously Slowing Down Approximation* (CSDA) *range* is used to calculate the range via the mass stopping power:

$$R_{CSDA} \equiv \int_0^{T_0} \left(\frac{dT}{\rho dx}\right)^{-1} dT$$

The pathlength is inversely proportional to the stopping power. Range straggling affects the pathlengths of the protons and is a result of stochastic variations in rates of energy loss.

## 2.2 Physics of x-ray production

To be able to understand the physics behind a CT scanner and simulate x-ray attenuation, the production of x-rays is an important step, and are elaborated here. It is based on *Chapter 9 of Radiological Physics and Radiation Dosimetry by Attix (2008)* [7] *and Chapter 4 of Handbook of Radiotherapy Physics (2007) [9]*.

### 2.2.1 Bremsstrahlung X-rays

X-rays used in a CT-unit is equivalent to bremsstrahlung produced in an x-ray tube. Inside the tube its vacuum, and by setting a voltage $V$ across the tube an electron beam is accelerated and allowed to strike a metallic target (the anode). The electrons are generated with a hot cathode. The photon energy spectrum will depend on the kinetic energy of the electrons which is dependent on the tube voltage (kinetic energy = e x V). A fraction of the kinetic energy of the electron will be transformed into bremsstrahlung x-rays, and the rest will be degraded to heat in the target. To convert a larger fraction of energy into x-rays a target with high atomic number should be chosen, but at the same time the target needs a high melting point. A common choice for the target is Tungsten (Z=74).

### 2.2.2 Kramer's spectrum

Kramer's spectrum is a simplification of the x-ray spectrum produced by bremsstrahlung and might be used in an x-ray spectrum simulation. As a criterion, the probability for an interaction is the same for all locations in figure 2.2. If the electron interacts close to the nucleus, as trajectory 1 in figure 2.2, its trajectory will be bent more than if it hit further out from the nucleus, trajectory 2 in figure 2.2.

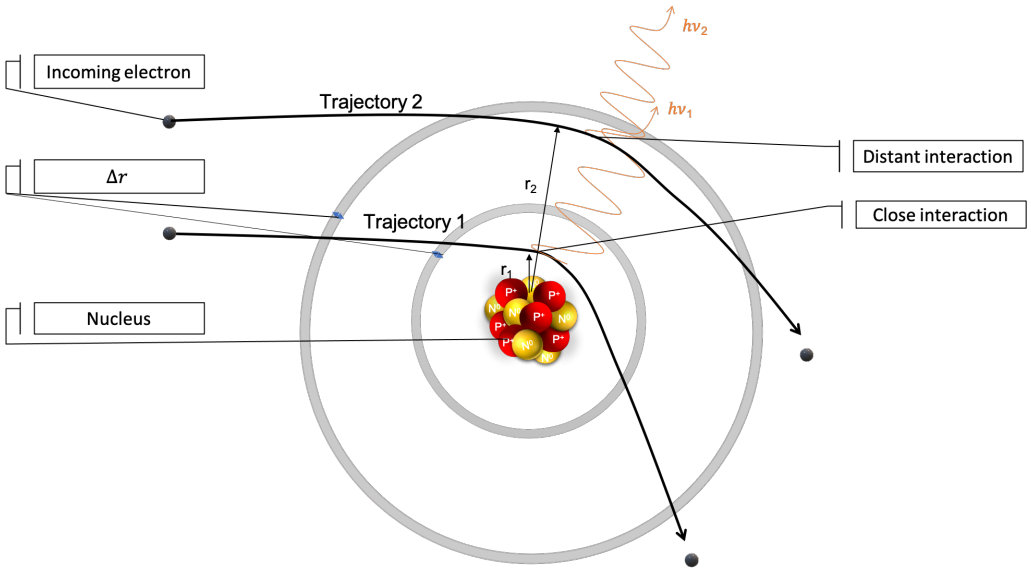


*Figure 2.2: X-ray production from bremsstrahlung. An incoming electron will be bent due Coulomb forces, and an emission of photons takes place. Here two incoming electrons is illustrated with trajectory 1 and 2. The photon energy will be larger for $h\nu_1$ than for $h\nu_2$.*

The location of the interaction give rise to the amount of energy transferred from the electron to bremsstrahlung x-ray, i.e., in the figure 2.2 $h\nu_1 > h\nu_2$. This gives the energy approximation:

$$h\nu \sim \frac{1}{r}$$

were $h\nu$ is the photon energy, and *r* is the distance from the nucleus. From this the ratio between x-ray energy in $r_1$ and $r_2$ becomes:

$$\frac{h\nu_1}{h\nu_2} = \frac{r_2}{r_1}$$

Furthermore, the number of electrons *N* that will interact with a small annulus at radius *r* with thickness $\Delta r$ is proportional to the area *A* of the annulus:

$$A = 2\pi r\, \Delta r$$

The energy fluence of bremsstrahlung photons generated from electrons passing a distance *r* from the nucleus is:

$$\Psi = N \cdot h\nu \approx N \cdot h\nu$$

Thus, the ratio of energy fluence at two distances is:

$$\frac{\Psi_1}{\Psi_2} = \frac{N_1 \cdot h\nu_1}{N_2 \cdot h\nu_2} = \frac{r_1}{r_2} \cdot \frac{r_2}{r_1} = constant$$

Thus, the x-ray energy fluence will be constant (i.e., the energy fluence spectrum is flat) when the electrons traverse an infinitely thin material. In a real situation, after the electrons have traversed the first layer of anode, the maximum kinetic energy of the electron will decreases, but the same procedure will repeat itself throughout the complete target. When adding up to the flat photon energy fluence spectrum for each layer of the anode, the total spectrum will thus take a linear form as in figure 2.3. This unfiltered spectrum is called *Kramer's spectrum* (Kramers, 1923) and is given by:

$$\Psi = K(h\nu_{max} - h\nu) \qquad \text{Eq. 2.4}$$

The different variables in equation 2.4 are described in table 2.2.

*Table 2.2: Parameters and definitions in Kramer's spectrum.*

| ***Symbol*** | ***Definition*** |
|---|---|
| Ψ | Photon energy fluence |
| $h\nu_{max}$ | Maximum photon energy, corresponding to the x-ray tube voltage |
| $h\nu$ | Energy of the photon |
| K | Constant |

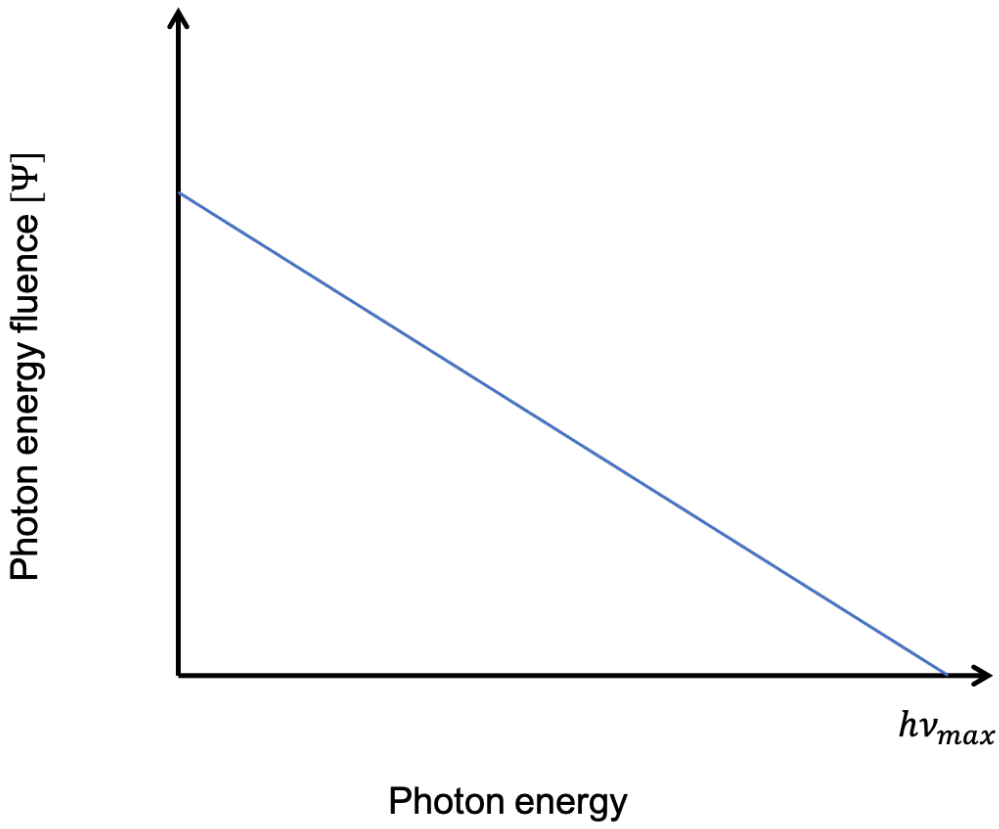


*Figure 2.3: Unfiltered energy spectrum produced by Kramer's law.*

In a clinical setting the photon beam is often filtered, and this will modify the photon energy fluence spectrum. In this case, the lower energies will be absorbed in the filter. The filter often consists of a material with a relatively high atomic number, and the photoelectric effect will be the main interaction for low energy photons (this will be further explained in chapter 2.3.2).

## 2.3 Photon interactions (uncharged ionizing radiation)

To produce medical images which will be used in this thesis, photons with energies of 120 kVp will be used. An introduction of photon interaction will be presented but due to low kVp energies pare production will not be mentioned. This chapter is based on *Chapter 3 and 7 of Radilogical Pyhsics and Radiation Dosimetry by Attix [7], part A Chapter 4 of Handbook of radiotherapy physics by Mayles et al [9].*

When an x-ray beam passes through a medium, an interaction between photons and a medium is possible, which results in energy transferred to the medium. The photons lose their total energy in relatively few large interactions. The probability of these interaction taking place is given by a concept called *attenuation* $\mu$, which is relevant primarily to these uncharged ionizing radiations. From the energy transfer between photons and medium an electron often is ejected, and the high-speed electron will transfer its energy by many ionizations and excitations of the atoms along its path. These ionizations and excitations may lead to detrimental effects if the medium is body tissue. The probability of interaction with a target entity is usually expressed in terms of the cross-section $\sigma$. The photon interaction is of interest since a photon beam traversing a patient gives rise to a CT image.

### 2.3.1 Attenuation

To explain the concept of attenuation, imagine that a monoenergetic parallel beam consisting of $N_0$ uncharged particles are incident perpendicularly on a flat plate of absorber with a thickness $l$. For an illustration see figure 2.4. At a fixed (long) distance on the opposite side of the absorber, a small detector is located. Only photons which have not interacted with the absorber (primary photons) are measured by the detector, the scattered photon will not reach the detector since the distance is long and the detector is small. Thus, if a photon interacts with the absorber it will either be absorbed or scattered away from the detector.

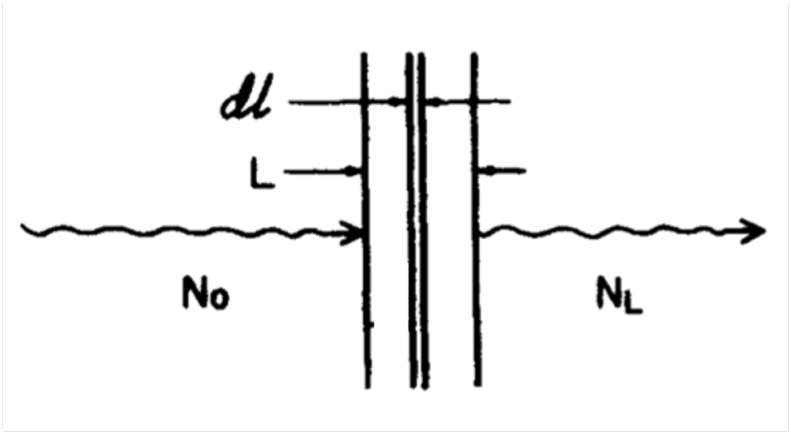


*Figure 2.4: Simple exponential attenuation through an absorber with thickness L. $N_0$ is the number of uncharged particles with a trajectory into the absorber, and N is the number of uncharged particles who manage to traverse the abosrober.*

Let $\mu \cdot 1$ be the probability that an individual particle (photon) interacts in a unit thickness absorber traversed. Then $\mu dl$ will be the probability that it will interact in an infinitesimal

thickness $dl$. The reduction in number of photons ($dN$) is proportional to the number of incident photons (N) times the thickness of the absorber ($dl$), mathematically:

$$dN \propto N\, dl$$
$$dN = -\mu \cdot N\, dl \qquad \text{Eq. 2.5}$$

where $\mu$ is a constant of proportionality called *attenuation coefficient*, and has the unit $cm^{-1}$ or $m^{-1}$, and dl is correspondingly in cm or m. The minus sign indicates that the number of photons decreases as the absorber thickness increases. Equation 2.5 can be written as:

$$\frac{dN}{N} = -\mu\, dl \qquad \text{Eq. 2.6}$$

Equation 2.6 gives the fractional change in N due to the absorption and scattering of photons in dl. To solve this differential equation an integration over the depth x from 0 to L, and the integration over particle population from $N_0$ to $N_L$, it gives:

$$\int_{N=N_0}^{N=N_L} \frac{dN}{N} = -\int_0^L \mu dl$$

$$\ln N_L - \ln N_0 = -\mu L$$

$$e^{\ln {N_L}/{N_0}} = e^{-\mu L}$$

$$\frac{N_L}{N_0} = e^{-\mu L}$$

$$N_L = N_0 e^{-\mu L} \qquad \text{Eq. 2.7}$$

This is the *law of exponential attenuation*. It is stressed that this only provides a measure of the number of primary (non-interacting) photons at a given depth.

### 2.3.2 Photoelectric effect

When a photon is absorbed by an atom it results in one of the orbital electrons to be ejected, which is called the photoelectric effect and is illustrated in figure 2.5. When one of the orbital electrons are ejected there will be a vacancy in the shell, this vacancy will be filled with an electron from another shell. In this process the atom will emit either characteristic x-ray or auger electron, depending on which shell the vacancy is in.

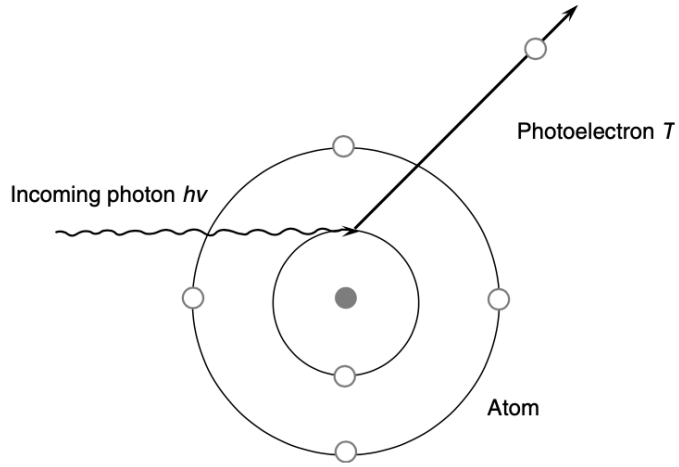


*Figure 2.5: Photon interaction with an atom, where the photon is absorbed and an electron is emitted. This interaction is called a photoelectric absorption and is mainly dominant for photons with low energy.*

When the photons interact with the atom via the photoelectric effect it will essentially transfer all of its energy to the atomic electron. The kinetic energy of the ejected electron is equal to $E_K = h\nu - E_B$, where $h\nu$ is the energy of the incident photon and $E_B$ is the binding energy of the electron. These types of interactions can take place with electrons in the K, L, M, or N shell. From Bohr's atom model we know that the binding energy of the electron depends on which shell the electron is located, and also what type of medium the target is.

For the photoelectric absorption the cross-section per atom, $\sigma_{pb,}$, depends strongly on the atomic number and photon energy, and it is approximately proportional as:

$$\frac{\sigma_{pb}}{\rho} \propto \left(\frac{Z}{h\nu}\right)^3$$

### 2.3.3 Compton scattering

In Compton scattering the photon interacts with a "free" atomic electron, in this case the binding energy of the electron will be much lower than the energy of the photon. The electron in this interaction will receive some energy from the photon and is emitted at an angle, while the photon will be scattered in another direction. See figure 2.6 for a schematic view of the interaction.

Since the Compton scattering involves a free electron in the absorbing medium, it is independent of the atomic number $Z$ for the cross-section per electron. This result in a proportional dependency for the cross-section per atom, $\sigma_c \propto Z$. From this it follows that the Compton mass attenuation coefficient ($^{\sigma_C}/_{\rho}$) is independent of Z and only depends on the number of electrons per gram. The number of electrons per gram is approximately the same for most materials except hydrogen.

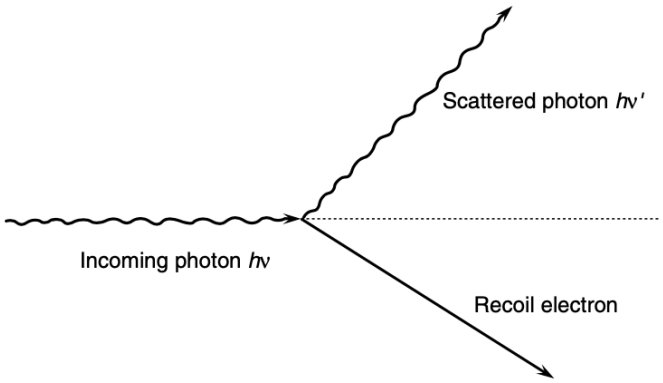


*Figure 2.6: An illustration of Compton scattering. Here a photon interacts with a "free" electron, this results in both particles being scattered in different angle.*

### 2.3.4 Coherent scattering – Rayleigh scattering

The incident beam of photons (electromagnetic wave) will, while passing close to the electron, set the electron into oscillation. The electron will oscillate at the same frequency as the electromagnetic wave and reradiate (emit) the energy in the same frequency as the original electromagnetic wave. This implies that no energy is changed into electronic motion and no energy is absorbed in the medium. The only result is the small angles of the scattered photons. Coherent scattering is most probable in high-atomic number materials and for photons of low energy. The mass attenuation for Rayleigh scattering is:

$$\frac{\sigma_R}{\rho} \propto \frac{Z}{(h\nu)^2}$$

## 2.4 CT

This chapter is based on *Computed Tomography by Kalender* [10] and *Webb's physics of medical imaging by Flower, M.A (2012)* [11]

*Computed tomography* (CT) scanner is a medical device to acquire images with information about the anatomy inside of the patient. The CT scanner consist of a gantry shaped as a doughnut, housing the appliances used to acquire the image and is illustrated in figure 2.7. A tabletop moves the patient through the gantry while the appliances is continuously rotating inside the gantry. The most important appliance in the gantry is the x-ray tube and detector, which generates and detect the photons. The tube and detector are placed perpendicular and moves continuously in the gantry. The acquired images are often used in medical situations as radiation therapy, for delineation but also for dose calculation.

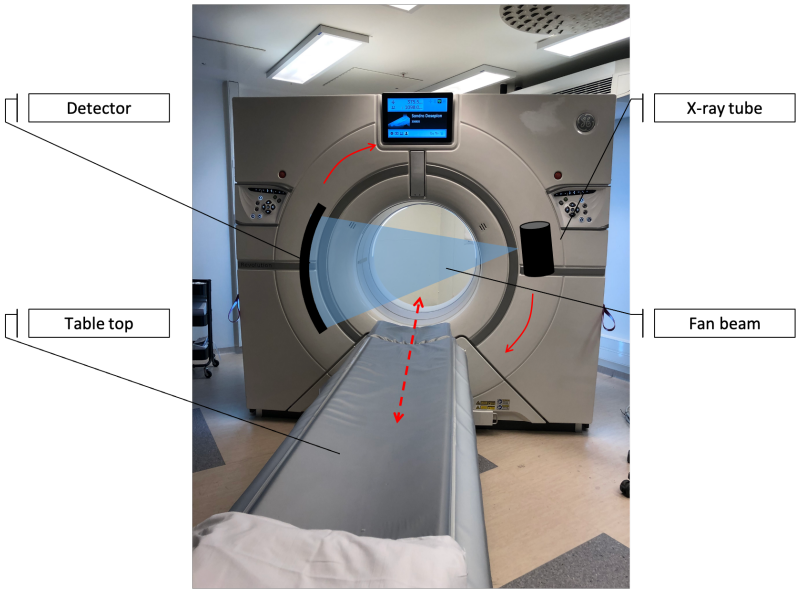


*Figure 2.7: A schematic view of a CT scanner. The gantry houses both the x-ray tube and the detector, which can be continuously rotated inside the gantry. The x-ray tube produces a fan shaped beam. In the CT scanner a tabletop which can position the patient into the gantry is an important part.*

An important difference between CT scanners used in radiotherapy and diagnostic is the tabletop. In radiotherapy one has to be able to reconstruct the CT image when the patient lays on the tabletop to receive therapy, due to this the tabletop in a CT for radiotherapy has to be flat, and not curved as in figure 2.7.

### 2.4.1 Basic principles of CT

When acquiring a CT image, the x-ray beam produces a range of photons with different energies to traverse through the patient. As the photon beam traverse the patient some of the photons might interact with the tissue and be absorbed, while other photons manage to traverse the patient without interaction. These photons, who has not interacted will be detected in the detector and contribute to a photon intensity. As the x-ray tube and detector is rotating the photon beam is continuously traversing the patient, as a result projections from each angle around the patient is obtained. The projections are used to reconstruct the attenuation coefficient, and to characterise what type of anatomical structure is in the patient.

Photon intensity gives the probability of traversing the patient without interacting, and this probability is dependent upon the sum of x-ray attenuating properties of all the tissues along the trajectory of the x-rays.

The basic principle behind a CT is measuring the spatial distribution of the attenuation properties from different directions and to compute superposition-free images from these data. In a CT the primary intensity $N_0$ and the detected intensity $N$ is known, from this the attenuation along the path. In chapter 2.3 attenuation was explained, and by rearranging equation 2.7 the equation for attenuation is given as:

$$\mu = \frac{1}{x} \cdot \ln\left(\frac{N_0}{N}\right) \quad \text{Eq. 2.8}$$

for a simple homogeneous object, with monochromatic radiation and is called an attenuation profile. In an inhomogeneous object the attenuation will depend on the different materials in the path of the x-ray. The linear attenuation coefficient is also dependent on photon energy, which may be a source to problems such as beam hardening. But it is also an advantage for *Dual Energy CT* (DECT), where material-selective measurements are done.

In the CT the projections are measured continuously and transferred to a data processing unit when the system is rotating around the patient. A CT scanner typically measure around 3500 projections with 600 – 1200 data points per projection.

From the projection data it is still unknown how the attenuation is distributed through the trajectory of the photon, and an inverse transformation has to be carried out to determine $\mu(x, y)$. Both filtered (convolution) back projection and iterative method are used for this purpose.

Filtered back projection use the obtained projections and add them to an empty image matrix along the angle which they were acquired to reconstruct the attenuation profile. While the main aim in iterative reconstruction is to adjust real time measurements into agreement with an image assumption made prehend.

### 2.4.2 Hounsfield Unit

The CT measures and computes the spatial distribution of the linear attenuation coefficient $\mu(x, y)$. But $\mu$ (physical parameter) is not very descriptive and is strongly dependent on the x-ray energy used, making a direct comparison of images obtained on scanners with different voltages and filtration limited. Therefore, the computed attenuation coefficient is displayed as a so-called CT value relative to the attenuation of water. This so-called CT value is named after the inventor and is specified in *Hounsfield units* (HU). For an arbitrary material *M* with attenuation $\mu_M$, the CT value is defined as:

$$CT\ value = \frac{\mu_M - \mu_{water}}{\mu_{water}} \cdot 1000\ HU \quad \text{Eq. 2.9}$$

From this equation water and air have the CT value 0 HU and -1000 HU respectively. These values are independent of the energy of the x-rays and therefore constitute the fixed points for the CT value scale. The Hounsfield scale do not have a lower limit, but for medical scanners it is normal to have -1024 HU and +3071 HU, respectively. If a digital bit resolution of 12 bits is used, this results in 4096 ($= 2^{12}$) different CT values.

### 2.4.3 Principles of sectional imaging

In a CT scanner the photon beam from the x-ray tube is shaped as a thin fan beam. This property of the beam makes it possible to acquire a CT image of a thin slice in the patient body. The photon beam will only pass in the direction that are contained within the plane of the slice, and at the same time no other part of the body outside the planar slice will be exposed by the primary photon beam. As a result of this a sectional image of the human anatomy with a spatial resolution of about 1 mm (chosen by the radiograph) and a density (linear attenuation coefficient) discrimination of better than 1% is made.

The continuously rotation inside the gantry of both the x-ray tube and the detector is a third-generation scanner, and is the only type of scanner that will be used in this thesis. The fan beam has a typically field of view of 50 cm, which the entire detector covers. Together with the moveable tabletop it is possible to continuously acquire CT images as the patient is translated through the scan plane. This is called a spiral mode, but also known as a helical mode due to the shape of the path around the patient.

### 2.4.4 Image quality

A quantitative determination of the image quality will make the process of comparing images easier. In this thesis the following quantitative metrics will be discussed as they are the most important [12].

*Image noise* is defined as fluctuation of CT numbers around a mean CT number in a CT image. This can be expressed with standard deviation $\sigma$:

$$Noise \propto \sigma = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}} \qquad \text{Eq 2.10}$$

and the different variables are defined in table 2.3.

*Table 2.3: Defining the variables used in equation 2.10*

| *Symbol* | *Definition* |
|---|---|
| $x_i$ | CT number for a pixel i |
| $\bar{x}$ | Mean CT number of the pixels |
| n | Total number of pixels |

This property is the magnitude of image noise, but do not describe the frequency content of the noise.

*Contrast to noise ratio* (CNR) can be defined as the difference in CT number between the object $\bar{x}_O$, and the background $\bar{x}_b$, divided by the noise $\sigma$:

$$CNR = \frac{\bar{x}_O - \bar{x}_b}{\sigma} \quad \text{Eq. 2.11}$$

A parameter that affects the noise in a CT image is how many photons are traversing the medium, *field of view* (FOV). The number of photons traversing material increases with higher current [mA], and a higher number of photons will reduce the noise in the images. Also, the pitch contributes to the number of photons traversing the medium, for q < 1 the projections will overlap, which mean a higher number of photons will travers the medium than if q > 1. The pitch is defined as:

$$q = \frac{T}{b} \quad \text{Eq. 2.12}$$

As seen, it depends on both the table increment *T* and the slice thickness *b*.

The photon beam traversing the patient will contribute to a dose delivery in the patient, this amount is much lower than the dose from the radiation therapy. From a clinical perspective this is not an important factor, but it will give some information about the quality of the CT image. The total dose to the volume is measured in *Computed Tomography Dose Index Volume* ($CTDI_{vol}$) and will be given for the phantoms used in this thesis.

$$CTDI_{vol} = \frac{1}{p}\left(\frac{1}{3}CTDI_{100,c} + \frac{2}{3}CTDI_{100,p}\right)$$

A CT image is a 512x512 image matrix with a voxel size depending on the *field of view* (FOV), matrix size and the slice thickness. A large voxel size reduces the noise but makes it harder to differentiate the details. Whereas a small voxel size will increase the noise but give a highly detailed image. The reason for the increased noise is the lower number of photons in each voxel. This gives that a larger FOV will give a higher voxel size since

$$Voxels\ size = \frac{FOV}{Matrix\ size}$$

## 2.5 Dual Energy Computed Tomography (DECT)

This chapter is based on *Spectral Computed Tomography by Heismann et al (2012) [13]* and *Dual energy CT in oncology by De Cecco et al (2015) [14].*

In a *dual energy CT* (DECT) two acquisitions with different energy (kVp) is acquired, see figure 2.8 for a schematic illustration. The motivation behind DECT is that the two radiation beams are attenuated differently through the body, and by combining information from these two images, a reconstruction of other image types is possible. In this thesis the focus is on

monochromatic (keV) images since we want to see how the HU values changes for monochromatic images compared to the HU values in a conventional CT image.

A conventional CT acquires an image of an area in a few seconds, meaning an image of linear attenuation throughout the structure. Today a 120 kVp CT image is the primary used dataset in treatment planning for radiotherapy. One of the challenges with CT is that it only reveals the patient´s morphology, but no information about the chemical composition which is important part of proton treatment plan.

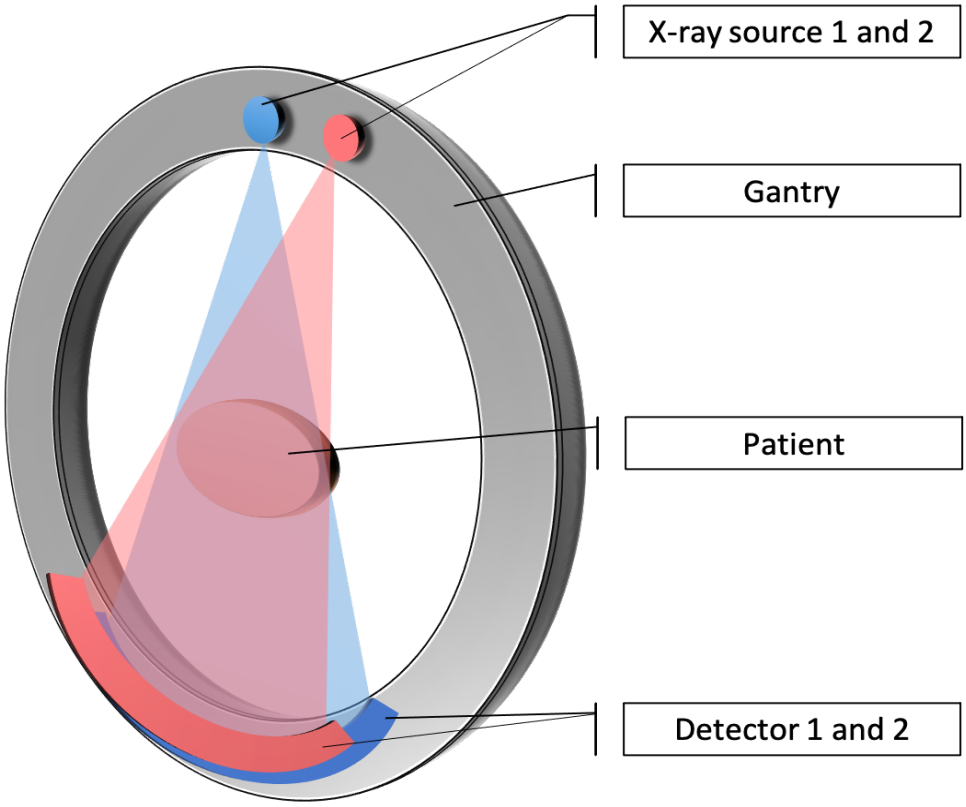


*Figure 2.8: Schematic view of a DECT scanner. Two x-ray sources and two detectors perpendicular to each other, are rotating simultaneously inside the gantry. The patient is located in the centre of the gantry.*

### 2.5.1 Basic

The x-ray attenuation through the material depends mainly on three different parameters, *type of material*, *density*, and *x-ray spectrum*. To obtain best possible spectral separation one should use spaced kVp settings, most commonly 80 kVp and 140 kVp is used since this typically will give best spectral separation and is available on most DECT machines. Today it is also possible to use 70 kVp on some scanners, and for an even better separation a tin filter is used on the high energies. For low energies the photoelectric absorption will be

predominant, but for higher energies the Compton scattering will be predominant. This results in an attenuation combined of these two interactions:

$$\mu(E) = \mu_C(E) + \mu_{pb}(E)$$

which can help characterise the material composition of the tissue. Moreover, two-material decomposition images can be obtained, which will be used to produce monochromatic images. The basic assumption of the two-material decomposition is that the mass attenuation $\frac{\mu}{\rho}$ of all materials can be expressed with sufficient accuracy as a linear combination of the photoelectric and Compton attenuation coefficients. As a consequence, $\frac{\mu}{\rho}$ of any material can be expressed as a linear combination of $\frac{\mu}{\rho}$ of two basis materials, where both materials differ in their photoelectric and Compton characteristics. An appropriate selection of materials is recommended, to minimize noise amplification. Ideally, one material will have a strong dependence on the photoelectric effect (e.g., calcium or iodine) whereas other material has a strong dependence on the Compton scattering (e.g., water). The attenuation is energy dependent, and by use of different values of the x-ray tube voltage, the energy spectrum of the x-rays will be different. Also, the x-ray quanta will interact with matter in different ways.

In CT, each ray (represented by the line integral L) from the x-ray source to the respective detector element can therefore be expressed as a linear combination:

$$L(E) = \left(\frac{\mu}{\rho}\right)_1 (E) \cdot \rho_1 \cdot t_1 + \left(\frac{\mu}{\rho}\right)_2 (E) \cdot \rho_2 \cdot t_2 \qquad \text{Eq. 2.13}$$

Where $\frac{\mu}{\rho}$ is the energy absorption coefficient, and $\rho_1 \cdot t_1$ and $\rho_2 \cdot t_2$ are the product of density and thickness of materials 1 and 2, respectively. If one measures L at two different energies, the above equation can be solved, and information about density and thickness for each voxel or area of interest in the CT images can be extracted.

### 2.5.2 Rapid switching of the x-ray tube voltage during the scan

In this thesis a GE revolution DECT scanner was used, and this machine offer DECT images by rapidly switching of the x-ray tube voltage, hence this is the method for DECT that will be explained.

By rapidly switching of the x-ray tube voltage between consecutive projections of the same spiral scan, the DECT dataset is obtained. As the DECT scanner is continuously rotating, the voltage over the x-ray tube will be switched, this will result in every other projection will be

for low/high kVp. To restore two images interpolation between the acquired projections are used. With this type of DECT the data are acquired in the full *scan field of view* (SFOV) of typical 50 cm is used.

Because of the nearly simultaneous acquisition of low and high energy, registration problems due to organ motion or contrast agent dynamics are small. However, the switching time between low and high kVp with the present x-ray tube technology is in the order of one-half millisecond. To avoid angular sampling artifact, one has to acquire a sufficient number of projections of both kVp settings (> 600 projections). Therefore, the fast kVp switching will be limited to a slower rotation time, about 0.3 – 0.8 s, even though the CT system may rotate faster in a non-dual-energy mode. As a result, rapidly moving organs, such as the heart, is challenging to scan with these systems.

While the x-ray tube voltage is rapidly switched, it is technically difficult to switch the x-ray tube current simultaneously. As a result of the equal x-ray tube current, the x-ray flux will be lower at 80 kVp than at 140 kVp and the dose will be different for the two values. A way to achieve equal dose for both energies, is by asymmetrical sampling. If the sampling of each 80 kVp projection is three times longer than the sampling time of the corresponding 140 kVp projection, or simply if about three consecutive projections are acquired at 80 kVp, while the next projection is acquired at 140 kVp. Then the x-ray flux is balanced, and no under- or overdose at either x-ray energy occurs.

Under- and overshoots of the x-ray tube voltage during the switching processes are a potential drawback of fast kVp switching. The actual tube voltage may not follow the ideal rectangular switching curve, as indicated in figure 2.9, but may show over- and undershoots. Consequently, the spectral separation and hence the potential or material discrimination might be reduced.

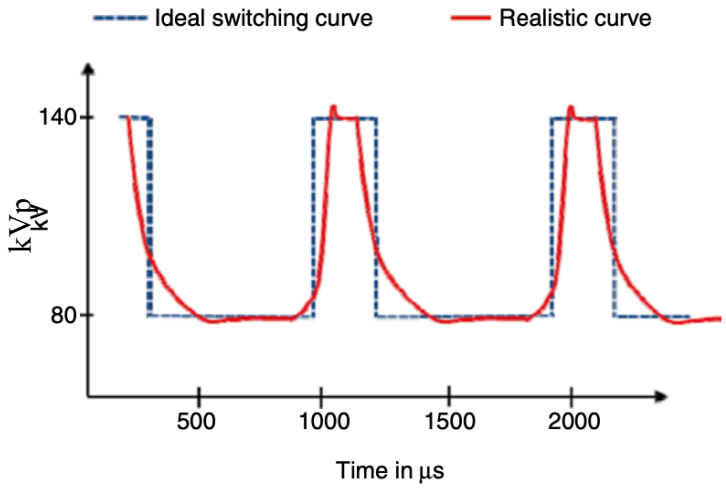


*Figure 2.9: Schematic view of rapid kVp switching between 80 kVp and 140 kVp. Note that to balance the radiation dose two or more low kVp projections are acquired for each high kVp projection.*

### 2.5.3 Monochromatic images

A monochromatic image, or a monoenergetic image is an image that is reconstructed to an arbitrary energy (keV). Monochromatic images can be derived using raw – data – based technique, this technique is based on the two-material decomposition (material specific image), where water and bone can be used as the basis material. Before actual processing of raw data can be performed, some prerequisite information must be made available, either by calculation from a model or from measurement. As a first step, for the two different spectra and different combinations of material thicknesses of the selected base materials, the respective line integrals L in equation 2.13 must be determined. As a second processing step, this information needs to be inverted so that information about the thickness of both materials is available. This must be done for each combination of line integrals measured at high and low voltages.

With these prerequisites fulfilled, calculation of pseudo-monochromatic images from the acquired high- and low-kVp data can be performed. The measured line integrals values $L_1$ and $L_2$ (respectively for 80 and 140 kVp) are converted to thickness values ($t_b$ and $t_w$) from a pair-wise lookup table, as shown in figure 2.10. Then a line integral L can be synthesized from thickness values using attenuation of both materials for the desired pseudo-monochromatic energy. Attenuation values can be extracted from standard tabulated data. The reconstruction of L finally allows for generation of monochromatic images at the desired energy level.

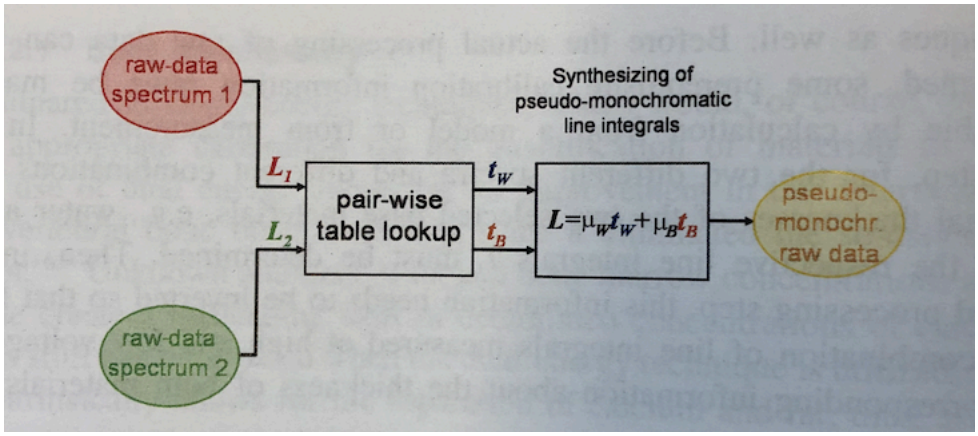


*Figure 2.10: Schematic view of reconstruction of monochromatic images*

The most obvious benefit of monochromatic imaging is the intrinsic reduction of beam-hardening artifacts and the elevated contrast for low energies, especially for examinations where iodine contrast is used.

## 2.6 Proton therapy treatment planning

This chapter is based on *Proton therapy by Yeung D. and J. Polta (2013)* [15], *Chapter 1 and 2 of Target Volume Delineation and treatment planning for particle therapy by Lee N. Y. et al. (2018) [16],* and *Proton beam Radiotherapy by Tsuboi and Gerelechuluun (2020) [17], otherwise referenced.*

### 2.6.1 The Bragg peak and Spread out Bragg peak (SOBP)

Proton therapy uses high-energy proton beams of energies up to about 200-250 MeV to irradiate cancers, and it is an alternative to the conventional photon radiation therapy. The advantage of proton therapy over photon therapy is mainly related to dose deposition in tissue. As the proton traverses and interact with the tissue it will slow down, and as the protons slows down it will use more time interacting with the molecules which leads to larger dose deposition towards the end of their range (cf equation 2.2). Therefore, the stopping power will increase with the depth of penetration. In the end of the range, the stopping power rises sharply which results in maximum energy transfer and dose deposition, called the Bragg peak. As seen in figure 2.11 this will result in a depth dose curve with low dose at the entrance, and almost no dose after the Bragg peak. On the other hand, a photon beam deposit maximum dose at 2-3 cm depth with a slow dose fall off deeper in the patient. This is clearly an advantage with proton treatment. However, the Bragg peak will be too narrow to cover a typical lesion. To be able to uniformly cover the lesion a Spread Out Bragg Peak (SOBP) should be used. To create a SOBP proton beam, different kinetic energies and beam intensity are used. This will result in a uniform dose over a larger longitudinal section. To be able to cover the lateral area of the lesion, either passive scattering or a scanning beam technique is used.

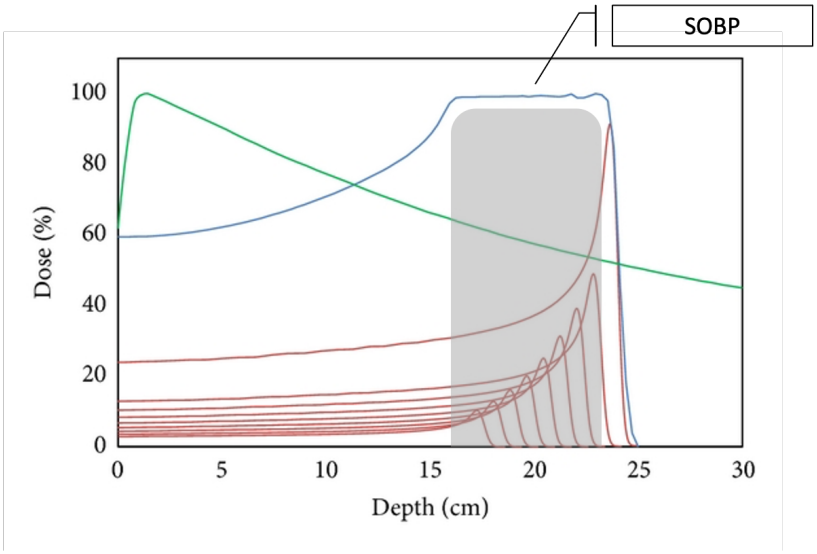


*Figure 2.11: Dose deposition for photons (green) and protons creating a SOBP (blue). Altered [18]*

### 2.6.2 Beam generator

To obtain a proton beam for clinical use the protons must be accelerated and either a synchrotron or a cyclotron is used for this purpose. The main difference between them is that the beam structure would be pulsed for the synchrotron and the output energy would be variable, but for the cyclotron the beam structure would be continuous and have a constant output energy. In this thesis only the principle of a cyclotron will be further described.

The cyclotron consists of two D-shaped electrodes named “Dee”, which is faced away from each other. Low-energy protons are injected into the centre of the cyclotron. The classical cyclotrons accelerate a proton in an electromagnetic field (RF field) between the Dee’s using an oscillating potential with frequency. The Dees are placed between magnetic poles, which will create a magnetic field perpendicular on the trajectory of the protons, which will ensure a spiral trajectory of the proton.

After acceleration the proton beam is guided through the beam transport system to the desired treatment room. It is common that one accelerator provides multiple rooms with protons. The transport system is a vacuum beam pipeline, which consists of a sequence of dipole magnets and quadruple magnets for steering and focusing of the proton beam.

Immediately after extracting the proton beam from the accelerator a degrader is used to modify the beam energy [19]. The proton beam is then bent into a gantry, which often provides 360° rotation from treatment delivery. The final element is the nozzle, which is located in the treatment room. There are two main nozzle types depending on which treatment modality is used (passive scattering and pencil beam scanning; see below). It is the nozzle that delivers the beam to the patient, but also it monitors the beam quality, alignment, and the dose delivery during treatment.

### 2.6.3 Treatment delivery

Proton treatment delivery is mainly divided into *Passive scattering* and *Pencil beam scanning* (PBS). Whereas passive scattering does not involve mechanical control during proton beam irradiation, pencil beam scattering uses a beam scanning method in which several devises are dynamically controlled during irradiation.

In passive scattering, a scatterer is used to obtain a beam with a large field size to cover the total target. The broad beam will be shaped with collimators as in conventional radiotherapy. To modulate the range according to the extension of the target a patient specific compensator has to be used. For best coverage of the target and to assure a minimum dose to healthy tissue.

The pencil beam scanning (PBS) method is a type of dynamic scanning and this is becoming the new standard technology in proton therapy [16]. The nozzle for a pencil beam scanning modality is illustrated in figure 2.12.

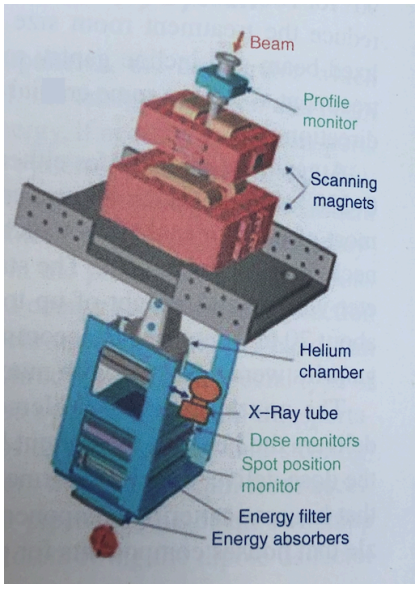


*Figure 2.12:: Schematic view of components on a nozzle for pencil beam scanning.*

In PBS the proton beam is monoenergetic, and two pair of orthogonal dipole magnets are used to steer the pencil beam as illustrated in figure 2.13. The pencil beam is moved left-right and up-down by the bending magnets and they will direct the pencil beam into a predetermined position with the desired intensity. When using PBS, the deepest layer in the target is irradiated first. As a result, the layers in front also get an amount of irradiation. So, when irradiating the next-deepest layer the irradiation dose should be reduced relative to a single irradiation dose. To reduce the irradiation dose the proton beam energy is degraded. In this way the total dose distribution will conform to a SOBP and a homogeneous dose will be given to the tumour. The pencil beam can either be continuous or discrete, but total dose to the lesion is the superposition of each individual pencil beam. Each proton beam used for total dose is shown as spots, as seen in figure 2.13.

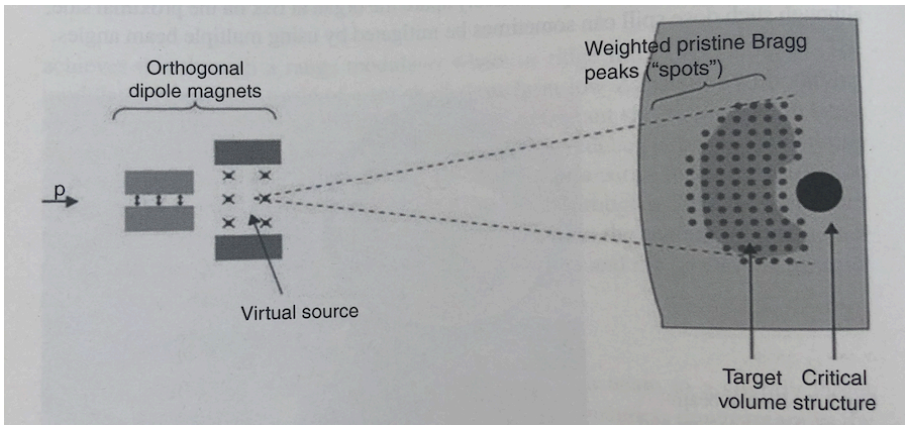


*Figure 2.13: Schematic view of dose delivery with pencil beam scanning modality. Here each pencil beam spot in one energy layer is illustrated around the target volume.*

The broad beam gives the passive scattering an advantage when it comes to patient movements, and beam stability is high in passive scattering relative to that in dynamic scanning. Typical parameters that affect the dose accuracy are the spot position, spot shape,

and the number of protons in each spot. For the scanning beam, it is very important to have a reliable and rapid-response control system that could deliver each spot to the desired position with the correct number of protons.

When a tumour is located close to the skin, an *energy absorber* can be used. This is because the lowest available energy from the accelerated proton beam may not be sufficiently low to guarantee target coverage. Sometimes it might be useful to use a collimating device for improved lateral beam penumbra [20].

### 2.6.4 Dose calculation

When the photon beam is traversing the patient, it will continuously lose energy. In the treatment planning system RayStation energy loss is calculated for each voxel. To lose a given energy $\Delta T$ in the patient *p* and water *w*, the protons need to traverse different pathlengths $\Delta Z$ and $\Delta Z_{eq}$, respectively. Furthermore, the energy loss is given by:

$$\Delta T = S_w \cdot \Delta Z_{eq}$$

$$\Delta T = S_p \cdot \Delta Z$$

where $S_p$ and $S_w$ is the stopping power in the patient and water, respectively. From this it finds:

$$\Delta Z_{eq} \cdot S_w = \Delta Z \cdot S_p$$
$$\Delta Z_{eq} = \frac{S_p}{S_w} \cdot \Delta Z$$

where $\frac{S_p}{S_w}$ is the *stopping power ratio* (SPR). Summing up over all steps $\Delta Z$ we find the pathlength in water, $Z_{eq}$, that is equivalent to the pathlength in tissue:

$$Z_{eq} = \sum \Delta Z_{eq} = \sum \frac{S_p}{S_w} \Delta Z$$

For an infinitesimal small voxel size $\mathrm{d}z'$ the sum becomes an integral over the total pathlength *z*:

$$Z_{eq} = \int_0^z \frac{S_p}{S_w} \, \mathrm{d}z'$$

$Z_{eq}$ is the equivalent depth in water at which a beam has lost the same amount of energy via electronic stopping as it has at the physical depth *Z* in the patient. In figure 2.14 the pathlength in water equivalent to the pathlength in a patient consisting of bone is illustrated.

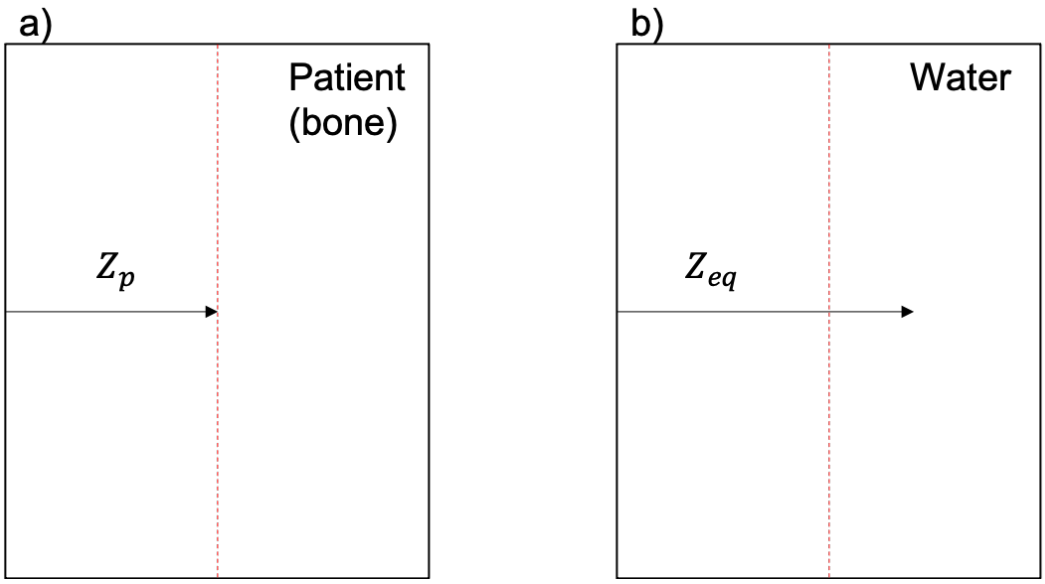


*Figure 2.14: a) A illustration of the wanted pathlength of a proton beam in a patient consisting of bone, b) the pathlength of the same beam but in water.*

As all dose measurements and calibrations of the dose calculations system is performed in water, this is the medium we want to relate all beam characteristics for. From a CT image of the patient, we know how to calculate the pathlength of the proton beam,but need to know what energy is needed to achieve $Z_{eq}$in water. When this is known the energy to reach the wanted pathlength in tissue is also known.

To obtain the SPR a conversion from HU to SPR is done. A table called *Heuristic Look Up Table* (HLUT) contains both the HU and SPR values for the same material, and to fill in the gap between known HU and SPR an interpolation is done. If there are multiple HU values for one material the SPR will change although it's the same material, this will lead to an error in calculation of the range dose. The table is not one straight linear fit, but a singularity is seen as shown in figure 2.15, which might increase the errors even more. The ultimate goal is independency from the HLUT, by instead sampling the electron density $\rho_e$ and stopping number directly from the DECT image.

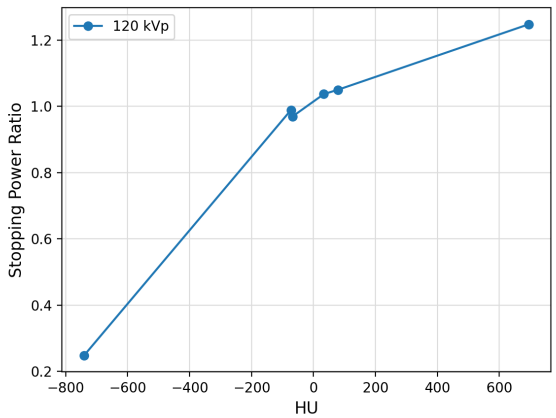


*Figure 2.15: Heuristic Look Up table (HLUT) for HU and SPR values.*

*The conversion of HU to SPR is called a CT calibration, and is a very important part of proton therapy.*

For a more accurate description of the passage of the beam spot in the patient, the spot is in the current treatment planning system RayStation, dived into 19 sub-spots, (figure 2.16). Along the true beam path through patient, pars of the spot may traverse e.g., bone or lung. The transport calculation system handles each sub-spot individually as a 'pencil beam', where each pencil beam has individual depth characteristics and broadening features. The total spot is then a superposition of all 19 spots [20].

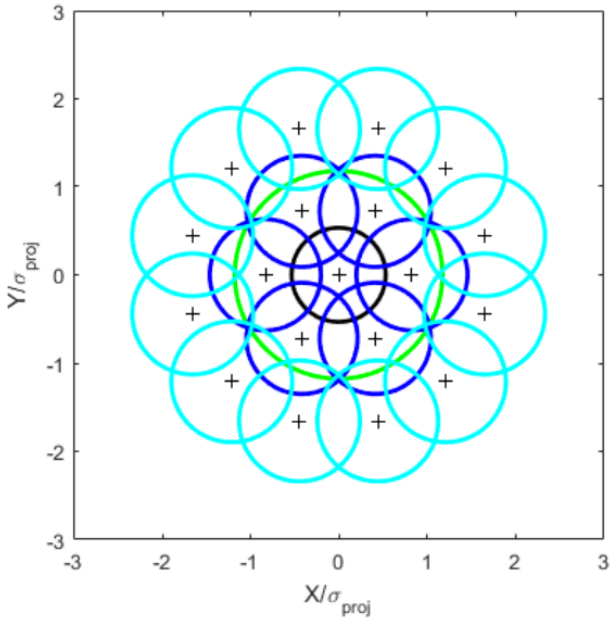


*Figure 2.16: Distribution of the 19 sub-spots (black, blue, and cyan) and the original spot (green) view from above [20].*

For the PBS two main approaches for dose delivery is mainly used. *Single field optimization* (SFO) or known as *Single field uniform dose* (SFUD), is a technique where each individual field uniformly cover a target. *Multi field optimisation* (MFO) or known as *intensity modulated proton treatment* (IMPT), where each individual fiend only partially covers a target, but the uniform target coverage is provided by the combination of al the fields included in the optimisation.

In proton therapy the *relative biological effectiveness* (RBE) of the proton beam is used to calculate the dose to the patient. Today a mean RBE value of 1.1 is the current standard defined by ICRU report 78[21]. In this thesis an $D_{RBE} = 1.1 \cdot D$ has been used for calculations.

### 2.6.5 Plan optimization

Inverse planning is an optimization technique that can be used for IMPT. It is used to find the best treatment plan. A set of dosimetric criteria for the treatment plan are defined, and the optimiser finds the best solution to the problem. The objective function can be defined as:

$$F(x) = \sum_{\sigma=1}^{S} \lambda_{\sigma}(D^{\sigma} - d^{\sigma})^2$$

$$D^{\sigma} = A^{\sigma}\, x, \quad \sigma = 1, \dots S; \quad x \geq 0,$$

Where $\sigma$ defines the structure (i.e., target volume or an OAR), $\lambda_{\sigma}$ is a structure-specific weighting factor, $D^{\sigma}$ is the calculated dose and $d^{\sigma}$ is the prescribed dose, $A^{\sigma}$ is the dose kernel matrix and $x$ is the intensity of the beam [22]. The optimal result is to minimise the objective function by finding the global minimum.

When defining the set of criteria as target volume or OAR, a weighting can be modified so the most important criteria, often target coverage, has highest weight. The number of iterations is either predetermined or set to run until the minimum is found. It is normal to have a large, predetermined value, so if the optimiser to not find a minimum value before this the optimisation is stopped and a final plan is constructed.

RayStation has developed a worst-case scenario approach to take the uncertainties into account, also referred to as minimax optimisation. The minimax optimisation aims at minimising the worst-case scenario for the objective function $f_i$, which gives a threshold for how much the treatment plan quality can deviate due to errors. Different types of errors may occur during treatment, they are divided into two main error sources, range, and setup uncertainties. If several errors are chosen, their weighted sum in the worst-case scenario will be considered. For the objective function $f_i$, who is required to be robust over the scenarios in a set $S$, and which have a nonnegative importance weights $w_i$, a formulation of the minimax optimisation problem can be:

$$\min_{x \in X} \max_{x \in S} \sum_{i=1}^{n} w_i\, f_i\big(\, d(x; s)\big)$$

Where $X$ is a set of feasible variables (spot weights for IMPT), and $d(x;s)$ is the dose distribution as a function of the variables $x$ and the scenario $s$.

The information about the uncertainties and errors are incorporated into the optimisation and it enables the treatment planning system to determine where to deposit dose to achieve plans that are robust against setup error and range uncertainties. Hence, the worst-case approach determines the IMPT plan that is as good as possible for the worst error scenario [23].

### 2.6.6 Treatment volumes

As in conventional radiotherapy, definition of the tumour, adjacent *organs at risk* (OAR), and other anatomical structures is an essential part of the proton therapy planning process. For defining these areas different types of advanced imaging systems are used. CT is often used due to its widespread availability and the possibility to also use it for electron density and stopping power estimates.

Gross tumour volume (GTV) is the tumour volume that is visible on the medical image. Cancer cells encompassing the GTV are not visible on the medical images, but an extent of the GTV called the clinical target volume (CTV) includes these cells. The CTV is usually obtained from an empirical margin added concentrically to the GTV, and it is imperative that the CTV receives the prescribed radiation dose. They are both delineated in figure 2.17. In the photon radiation an additional volume called planning target volume (PTV) is used to include for organ motion and uncertainties in the setup and treatment delivery. This margin is used mainly to ensure the prescribed dose to the CTV. But for proton treatment, robust planning is often used instead of the PT, and in this thesis a robust planning will be used instead of the PTV, which is also done in this thesis. These delineation are as defined by ICRU report 50 [24] and 62 [25].

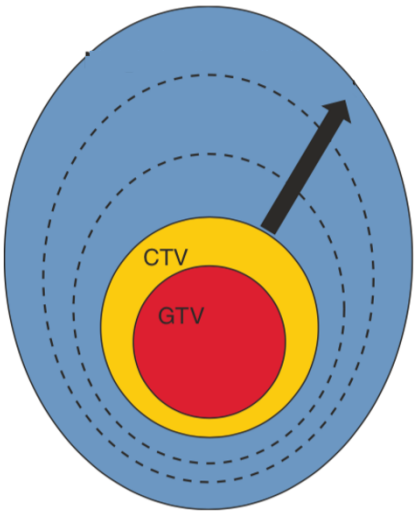


*Figure 2.17: View of gross tumour volume (GTV) and clinical target volume (CTV). A robust optimization is used to minimise the worst case of errors and the blue area around the GTV and CTV might be this robust area.*

It is not only the tumour within CTV that is important to delineate on the medical image, but also of great importance to delineate healthy and critical organs which is located close, or within the beam, referred to as OAR. For many of these organs tolerance dose levels have been established to avoid that its function will stop or weaken due to tissue damage.

# 3 Method

In this chapter the methods and materials to investigate the various aspects in this thesis will be introduced. The work is divided into two parts, where part 1 covers theoretical aspects and CT scanning of phantoms with known mass density, among other, while part 2 CT scanning of an anthropomorphic phantom and proton therapy planning.

## Phantoms with known mass density

1. •SECT and DECT scanning of two phantoms containing inserts with various known electron densities. Analysing the HU values from these scans (referred to as measured values). The vendor of one of the phantoms also gives the expected HU value for the inserts (reffered to as theoretical values)
2. •Calculating the attenuation and HU values from objects with the same electron density as in 1) in a simulated CT scanner (reffered to as simulated values)
3. •Calculating the SPR depending on the kinetic energy of the proton, mean exitation energy, and atomic numbe-of-mass.

## Anthropomorphic phantom

1. • SECT and DECT scanning of anthropomorphic phantom. Quantified the impact of this on two clinical plans. And see how HU changes with location in the image slize.

## 3.1 Phantoms with known density

### 3.1.1 Phantoms

Two phantoms were used in this thesis *Gammex – Multi energy CT - phantom* and *Quasar body phantom [26, 27]*. Both phantoms have inserts with different material composition.

#### 3.1.1.1 Quasar Body Phantom

The Quasar body phantom is illustrated to the left in figure 3.1 and it consist of an acrylic component shaped as the area of the abdomen of a person, width is 30 cm, height 20 cm and length 12 cm. Inside the acrylic component there are three cylindrical cavities where different types of inserts with a radius of 8 cm can be placed.

In the central cavity a cylinder with five inserts can be positioned. The five inserts have a radius of 3 cm and the thickness of 2 cm and are located as shown in the right panel of figure 3.1. All of the inserts are made of known material, and a QA is found on their webpage [26]

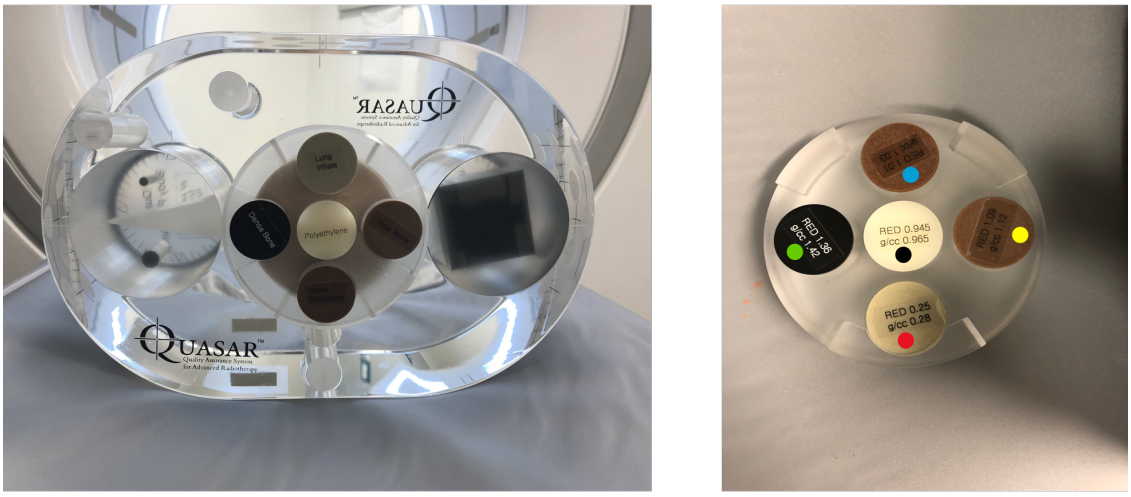


*Figure 3.1: To the left the Quasar phantom with the five inserts is shown. On the left a close up image of the 5 insert are shown. The red dot shows the 'Lung Inhale', the yellow shows the 'Inner bone', the blue dot shows the 'Water Equivalent', the green dot shows the 'Dense bone', the black shows the 'Polyethylene'. In the right image the holder with the inserts is rotated 180°.*

The five different inserts are as follow:

- Lung Inhale
- Dense Bone
- Water Equivalent
- Inner bone
- Polyethylene

In table 3.1 the mass density $\rho$ of each insert are listed, the values are obtained from the *Strålevernet (DSA)* [28].

*Table 3.1: Mass densities for the five different inserts in the Quasar phantom.*

| ***Material*** | ***ρ [g/cm³]*** |
|---|---|
| Lung | 0.28 |
| Dense bone | 1.42 |
| Water Equivalent | 1.03 |
| Inner bone | 1.12 |
| Polyethylene | 0.965 |

#### 3.1.1.2 Gammex – Multi-Energy CT – phantom

The Gammex Multi-Energy CT Phantom (figure 3.2) is made of a water-equivalent base material and contains cavities that can be fitted with rods of specific materials of interest. The phantom is shaped as a body and is approximately 40x30 cm in-plane and 16.5 cm depth. The cavities radius is 2.85 cm.

The phantom has 19 rods and from these this thesis will look at the Hounsfield Unit for six of them;

- High-Equivalency (HE) blood 70, mass density $\rho = 1.07\ g/cm^3$
- High-Equivalency (HE) blood 40, mass density $\rho = 1.03\ g/cm^3$
- High-Equivalency (HE) brain, mass density $\rho = 1.02\ g/cm^3$
- CT High-Equivalency Solid Water, mass density $\rho = 1.00\ g/cm^3$
- High-Equivalency (HE) blood 100, mass density $\rho = 1.10\ g/cm^3$
- High-Equivalency (HE) adipose, mass density $\rho = 0.94\ \ g/cm^3$


*Figure 3.2: Gammex phantom where the cavities are filled with the rods. The rods that are circled around is the rods that is for HU analyses in this thesis. They are of material 1) HE Blood 40, 2) HE Blood 70, 3) CT HE Solid water, 4) HE Brain, 5) HE General Adipose, and 6) HE Blood 100. (HE: High-equivalency)*

From the manual [29] the Hounsfield Unit for the different material – and different mono chromatic DECT image – is obtained, see table 3.2.

*Table 3.2: Hounsfield Unit for the different materials in the rods at different energies inserted to the Gammex phantom. The values are obtained from the manual [29].*

| ***Energies [keV]*** | ***HE blood70*** | ***HE blood40*** | ***HE brain*** | ***Solid water*** | ***HE blood100*** | ***HE adipose*** |
|---|---|---|---|---|---|---|
| 40 | 90 | 65 | 49 | -2 | 113 | -140 |
| 50 | 82 | 53 | 39 | -3 | 109 | -107 |
| 60 | 76 | 45 | 32 | -4 | 106 | -90 |
| 70 | 72 | 38 | 28 | -5 | 104 | -80 |
| 80 | 71 | 36 | 26 | -5 | 103 | -73 |
| 90 | 70 | 35 | 25 | -5 | 102 | -69 |
| 100 | 69 | 35 | 24 | -5 | 102 | -66 |
| 110 | 69 | 34 | 23 | -5 | 101 | -65 |
| 120 | 68 | 33 | 23 | -6 | 101 | -63 |
| 130 | 68 | 33 | 22 | -6 | 101 | -62 |
| 140 | 67 | 33 | 22 | -6 | 101 | -62 |

## 3.1.2 CT scan of the phantoms

### 3.1.2.1 Goal

The goal of this part is to scan with a Dual Energy CT to produce monochromatic images with energies of 40 keV up to 140 keV (intervals of 10 keV) and convenctional CT images witch 120 kV voltage. This will be done for both phantoms. Another aim is to reconstruct the image with two different methods, filtered back projection and iterative reconstruction.

### 3.1.2.2 CT acquisition settings

In this thesis a GE Revolution CT was used to acquire both DECT and SECT images. In table 3.3 the FOV, tube current, rotation time, and pitch used for the DECT are listed, resulting equal $CTVI_{vol}$ for the phantoms. In the DECT scan a voltage of 80 kVp and 140 kVp were used, and in the conventional single energy CT (SECT) a voltage of 120 kVp was used.

*Table 3.3: FOV, current, rotation time, pitch, and $CTDI_{vol}$ are values that were set in the DECT.*

| ***Phantom*** | ***FOV [cm]*** | ***Current [mA]*** | ***Rotation time [s]*** | ***Pitch*** | ***$CTDI_{vol}$ [mGy]*** |
|---|---|---|---|---|---|
| Quasar | 36 | 445 | 0.5 | 0.5 | 21.8 |
| Gammex | 41 | 445 | 0.5 | 0.5 | 21.8 |

For the SECT image the values had to be changed to reproduce an $CTDI_{vol}$, see table 3.4.

*Table 3.4: FOV, current, rotation time, pith, $CTDI_{vol}$, and Energy are values that were set in the conventional single energy CT (SECT).*

| ***Phantom*** | ***FOV [cm]*** | ***Current [mA]*** | ***Rotation time [s]*** | ***Pitch*** | ***$CTDI_{vol}$ [mGy]*** | ***Energy [kVp]*** |
|---|---|---|---|---|---|---|
| Quasar | 36 | 330 | 0.5 | 0.5 | 21.9 | 120 |
| Gammex | 41 | 330 | 0.5 | 0.5 | 21.9 | 120 |

Spiral scanning was used to produce the images for this thesis.

### 3.1.2.3 CT acquisition procedure

First step for a DECT scan protocol was to choose “default setting” for the Abdomen, due to the resemblance between the phantoms and an abdomen. A protocol to produce the monochromatic images for energies 40 – 140 keV, with both iterative reconstruction and back projection reconstruction, were made. The GE Revolution used 80 kVp and 140 kVp to

reconstruct the monochromatic images, see chapter 2.5.3 for more information about monochromatic images.

To be able to reconstruct the images with both back projection and iterative reconstruction there was an algorithm called ASiR-V that could be changed. For the back projection reconstruction ASiR-V equal to 0 % was chosen, and for iterative reconstruction ASiR-V equal to 50% was chosen.

A slice thickness to 1.25 mm was used, because the typical slice thickness on the CT images used in radiation therapy is between 1.00 mm and 2.00 mm. 1.25 mm would give the best images on this CT.

For these scans it was chosen to have the same rotation time (0.5s) and pitch (0.5). The field of view (FOV) was also kept constant for the different phantoms, Quasar (36 cm) and Gammex (41 cm). To be able to get a similar dose ($CTDI_{vol}$) for both the DECT and SECT images, one had to alter the current from 445 mA for the DECT images to 330 mA for the SECT images.

For the DECT scan of the Quasar phantom, it was placed with the front facing out of the scanner. The laser system installed in the CT was used to fix the phantom in the centre and made it possible to position the phantom in the middle of the CT before acquisition. In figure 3.3 the laser system and positioning of the Quasar phantom is shown.

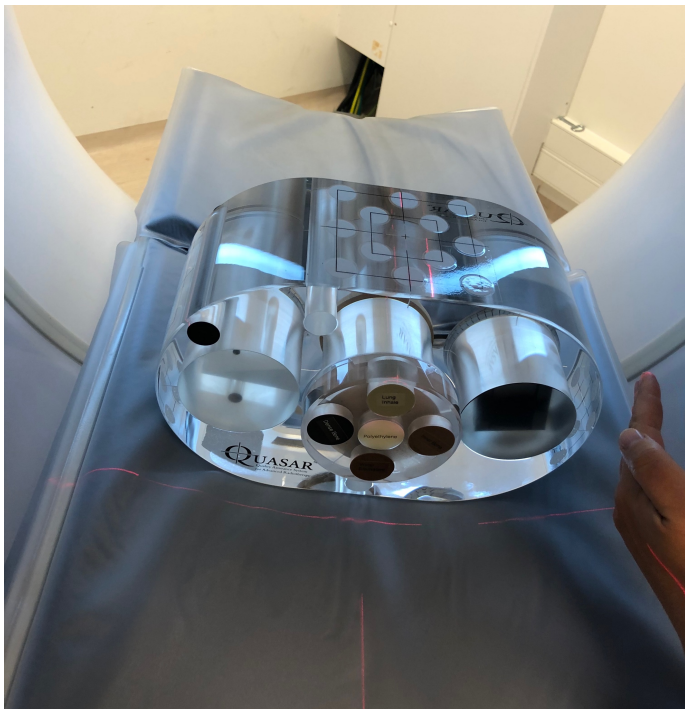


*Figure 3.3: An overview image from mounting the Quasar phantom in the gantry of the CT. The laser system was used to make sure the phantom was fixed in the centre of the gantry, and also to make sure the beginning of the phantom was in the middle.*

Then the Gammex phantom was placed on the coach and the laser system was used as just described, figure 3.4 shows this phantom positioned in the CT.

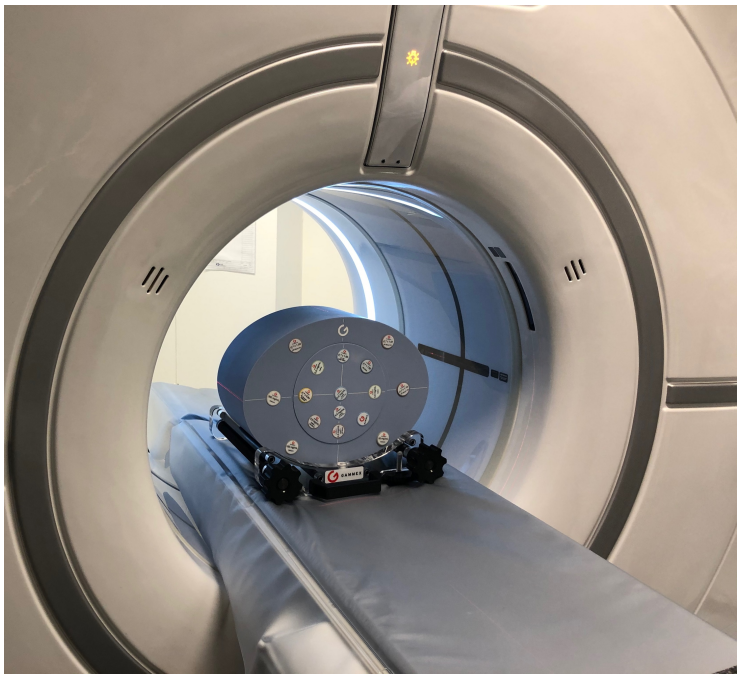

*Figure 3.4: The Gammex phantom fixed in the gantry of the CT. The laser system was used to make sure that the phantom was located in the centre of the machine before start.*

After the DECT scans the SECT scans was acquired. Since the Gammex phantom was already located in the gantry of the CT it was the first phantom to be scanned with SECT. Afterwards the Quasar phantom had to be positioned on the coach once again and this might affect the position or length of the SECT scan.

In these acquisitions spiral scanning with “feet first”-orientation was used, which would result in a mirrored image importing them into python or image J, viewer with the possibility to choose image-orientation will not have this problem.

## 3.1.3 Find HU values

### 3.1.3.1 Goal

The main attempt with this code was to make two templates that mask predetermined regions of interest (ROI’s) on DICOM files. From this, the code would be able to extract the intensity in the pixels which are located inside the ROI’s, the intensity in the pixels gives information about the Hounsfield Unit.

#### 3.1.3.2 Load DICOM files

The DICOM files were sent to *Sectra IDS7 Oslo university hospital* to be able to download the data.

The first important step was to upload the DICOM files into a python program. To do this *pydicom and glob* libraries had to be imported.

The downloaded 3D image data was not organized by slice number, so the code had to be able to:

1. Find the slice location (at what position in the image stack the image file was located).
2. Use pydicom and numpy's argsort to sort the image files from lowest to highest using the list of corresponding slice location.

The DICOM files used were of size [512,512], number of pixels in the x- and y-dimension, the number of slices gave the z-dimension.

The information from the pixels is not in HU at first, so one must convert by:

$$Output\ units = m \cdot SV + b$$

the variables and definition are listed in table 3.5.

*Table 3.5: Variables and definition of the equation above.*

| ***Symbol*** | ***Definition*** |
|---|---|
| Output units | Hounsfield Unit |
| m | Rescale slope |
| SV | Stored image values |
| b | Rescale Intercept |

From the DICOM files, m = 1 and b = -1024.

For a good estimation of the HU value of the ROI's, it was preferred to take the mean HU over a volume, and not only one slice. Here it was chosen to take the mean HU over 5 slices, this will be approximately $1.25\ mm \cdot 5 = 6.25\ mm$ thickness of the phantom and insert. To find a region of the phantom where the inserts were homogeneous, a function to show every seventh file was produced.

```
80    def sample_stack(stack, rows=4, cols=4, start_with=1, show_every=7):
```

#### 3.1.3.3 Template

A template was made by masking the circular ROI´s for the two phantoms, the circular ROI´s were made by the output code below. To find the centre of the inserts in the phantoms both *Image J* and a preview of a DICOM slice with masked ROI was used.

```
137      # create mask in a circular pattern around the centrum of a chosen index
138    for i in range(-ra, ra+1):
139        for j in range(-ra, ra+1):
140            r = np.sqrt( (i**2.0 ) + (j**2.0) )
141
142                if r <= ra:
143                  #fill the indexes that is a part of the circle with 1. (array)
144                    circle1[i + y1, j + x1] = 1
```

With Image J it was possible to figure out where the middle of the inserts was located, this information was then used to put a circular mask over the insert in the DICOM file. To make sure that the ROI was correctly located one would make a preview of the file. After making sure of this, one would store the indexes inside the position of the ROI´s. This way a template was made, and ready to be used on the rest of the images.

### 3.1.4 CT simulation in python

#### 3.1.4.1 Goal

To calculate the simulated HU values and understand how the relation between HU and SPR appears, a code in python was developed. The code calculated the transmitted x-ray spectrum through an object containing different materials via the exponential law of attenuation. Thereafter, the code calculated the HU value. The program also calculated the mass stopping power for protons with different kinetic energy penetrating for various materials. Finally, the code provided the relationship between the calculated HU and the Stopping power (SP). The calculated HU values from the program were compared to experimental values from the clinical CT scans.

#### 3.1.4.2 CT simulation

A CT scanner uses x-rays generated at rather low voltages (kVp) to acquire an image of an object. To simulate a photon beam from CT, Kramer's spectrum was used. This provided a photon beam with energies from 10 keV up to a maximum energy ($h\nu_{max}$) which was set to 120 kVp. The production of a photon beam is described in chapter 2.2. Below a snippet from the code on how the beam was created, based on equation 2.4.

```
14      hv      = 10 + np.arange(kv - 9)
…
21      # Kramers spectrum, info about how the x-ray spectrum from bremsstrahlung
22      psi_spec_CT = K * (hv_max - hv)
```

#### 3.1.4.3 Mean attenuation value

When the simulated photon beam traversed a material (absorbing medium or filter) the law of exponential attenuation (equation 2.7) was used, it is described in chapter 2.3. *National Institute of Standards and Technology (*NIST) has a website that contains attenuation coefficients and densities for different materials for a range of photon energies, they are obtained from *Table 4 [30]*. By downloading the tables from their site, it was possible to extract the attenuation coefficient for specific photon energies. A Scipy interpolation was used to extract the attenuation for an arbitrary photon energy *f*, see the "snippet" for python command. The interpolate function took the energy *x1* and attenuation coefficient *x2* as input variables.

```
114     f       = interpolate.interp1d(x1, y1, kind='linear')                   # interpolation
```

As in a real CT the photon beam in the simulation CT was sent through a filter before traversing the patient. This would give a beam hardening effect since most of the photons with lower energy was absorbed by the filter. By varying the filter (thickness and material) the attenuation would change. Since the photoelectric effect increases with atomic number Z, high Z-filters will be more efficient in filtering the beam. In this simulation an Aluminium filter with thickness 0.039 cm was used, which is the same as in the relevant CT scanner.

The code was able to distinguish between a monochromatic beam ([keV]) and a regular x-ray beam (Kramer spectrum). This made it possible to calculate the HU value for both a convetional CT image and monochromatic images reconstructed from DECT.

The function *filtration* was the main function in the code, here the mean attenuation through the absorbing medium for a given energy was calculated. As shown below, *filtration* was a function of the extracted attenuation, the type of filter used, the thickness of the filter, the energy of the photons (both in monochromatic and kVp), and what type of energy to expect (single energy or a range of energies), denoted *interp_mu, filt, thickness, energy, and typ,* respectievely. If typ is set to 0, an energy range is chosen, and for any other number a single energy is chosen.

```
147     def filtration(interp_mu, filt, thickness, energy, typ):
```

In this function, the total number of photons before and after penetrating the absorbing material is calculated. To do this the function *pre_filtration* is called on. In the *pre_filtration* function the energy range (Kramers law) or single energy is chosen and then filtered through

the Aluminium filter before both the filtered and non-filtered energy is returned to the main function.

The program simulates beam traversing through the absorbing material for different energies, here the variable interp_mu was used to inform about material type and the precalculated attenuation coefficient. Afterwards the total number of photons before and after traversing the absorbing medium was calculated. To do this the numpy function *sum* is used (see output below), this was equivalent to integrating over the whole energy range.

By using the information about the spectrum before and after traversing the absorbing material, it was possible to calculate the mean attenuation by using a form of the exponential law of attenuation (equation 2.7)

$$\bar{\mu} = \frac{\ln\left(\frac{\psi}{\psi_{after}}\right)}{x}$$

Mean attenuation depended on the thickness of the absorbing material, *x*, and the number of photons before and after traversing the absorbing material. For more information about Kramer's spectrum see section 2.2.1. In line 161 in the output from the code, this calculation is illustrated.

```
158    psi_interp_tot          = np.sum(psi_interp[1])
159    psi_interp_filter_tot           = np.sum(psi_interp_filter)
160
161    mu_mean_interp = (np.log( psi_interp_tot / psi_interp_filter_tot )) / x
```

This straightforward code would calculate the mean attenuation when a range of energy is used via Kramer's spectrum. For the mean attenuation for a single energy a range from 10 – 140 keV at 10 keV increments was used. In the output it is shown how the energy will vary between 40 – 140 keV, and how it was used to obtain the mean attenuation. The mean attenuation for a given energy is appended to a list, so it is possible to use the data later.

```
250        for i in range(10,140,10):
251            en = 10 + i
252            fil_soft  = read_file(Soft_t_file, Soft_t_dens)
253            inte_soft = interpolation(fil_soft, en)
254            mu_soft = filtration(inte_soft, element(Al_file, Al_dens), x_filter, en, 1)
255
256            mu_soft_list.append(mu_soft[4])
```

When the mean attenuation value was calculated it was possible to calculate the HU value, equation 2.9. The data generated for both monochromatic beams and regular beams is later used to compare the simulation with a real CT scanner.

### 3.1.5 Calculate Stopping power ratio

#### 3.1.5.1 Stopping power

The function *dedx* based on equation 2.2 was used to calculate the mass stopping power in the same material as used in the CT simulation. From a clinical point of view the kinetic energy of a proton is between 1 MeV – 250 MeV. The function takes *T, I,* and *Z_A* as variables representing the *kinetic energy of the proton, mean excitation energy,* and *atomic number-to-mass*, respectively. The variables used in the function are found on NIST webpage [30]. The program could calculate the mass stopping power for wanted materials, later on it was converted into stopping power by a multiplying with the mass density of the absorbing medium, and in the end stopping power ratio by dividing the SP for a medium with the SP for water. The function *beta* calculates the $\beta$ in the mass stopping power equation and returns it in the main function *dedx*.

```
643    def beta(T):
644            b = ( 1 - ( 1 / ( (T/M0_c) + 1  ) )**2 )**(1./2.)
645
646    return b
…
652    def dedx(T, I, Z_A):
653            be = beta(T)
654
655    # Stopping power with units [MeV/(g/cm^2)]
656            SP = 0.3071 * Z_A * z**2 * (1/be**2) * (13.8373 + np.log( be**2 / \
657                    ( 1- be**2)) - be**2 - np.log(I) ))
658
659            return SP
```

The data was used to convert HU values into SPR values. This would give an insight in how the SPR values might change as the HU values changes with energy, which could affect the dose calculation in proton therapy

## 3.2 Anthropomorphic phantom

### 3.2.1 CT scan

In order to further evaluate the clinical effect of applying DECT in proton therapy the Alderson radiation therapy phantom was used. The Alderson phantom is a human phantom which contains anatomical correct structures of a real human, a so the called anthropomorphic phantom. For the CT scan two different Alderson phantoms is used, one whole head and another in slabs. This was done to find out which of the two phantoms would be better to perform a proton dose plan on. Both are imaged so one later can choose which of them will give the best clinical information. The phantom contains the same anatomy as the human head, but it is missing the nasal cavities, and the eyes, and is also has a hole through the middle of the head. In figure 3.5 the phantom is displayed in a head holder.

To be able to make a proton plan and also be able to give the head phantom proton treatment, lead pellets were used to be able to reconstruct the positioning in the CT scanner. For a head phantom a lead pellet is placed in the forehead, straight line from the nose and over the eyes, and two additional lead pellets are located on both sides of the head. The laser build in the CT machine is used to place the lead pellet see figure 3.5 for an illustration of the location of the lead pellets.

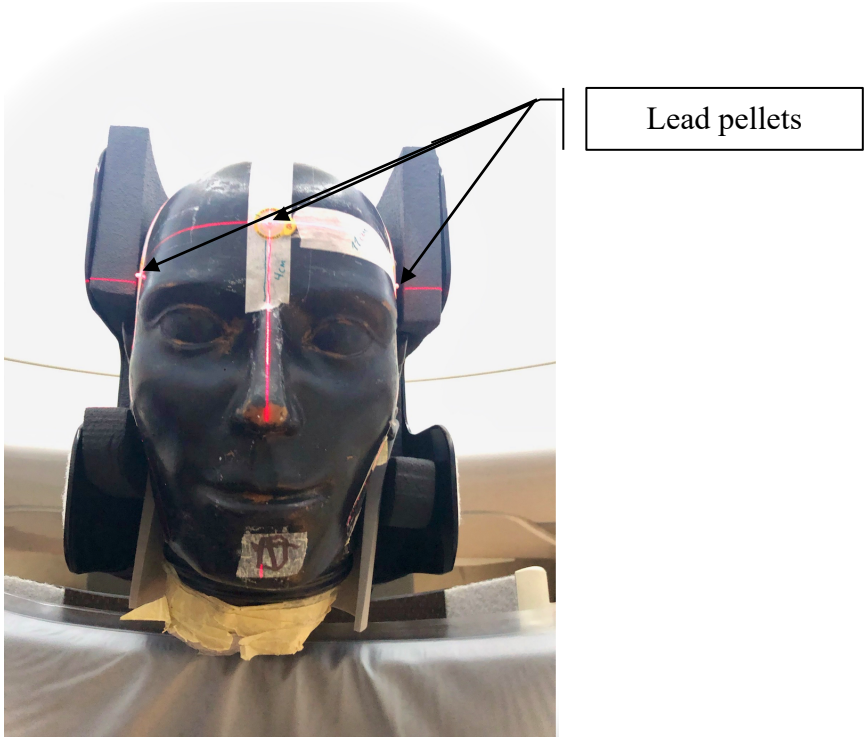


*Figure 3.5: Alderson phantom located in the DECT scanner. The arrows point on the lead pellets used for the treatment plan or if proton treatment were to be given to the phantom.*

For the Alderson phantom, head is set as the “default setting”, and mono chromatic images for 40-140 keV with both filtered back projection and iterative reconstruction was selected

from the DECT acquisition. Also, a CT scan with the same "default setting" with 120 kVp was used. When making the protocol for the CT image, an equal dose for both image sets is desired. From table 3.6 and 3.7 the different parameters used for the two scans are tabulated.

*Table 3.6: Parameters set for the DECT scan of the Alderson phantom.*

| ***Phantom*** | ***FOV [cm]*** | ***Current [mA]*** | ***Rotation time [s]*** | ***Pitch*** | ***$CTDI_{vol}$ [mGy]*** |
|---|---|---|---|---|---|
| Alderson | 24.8 | 315 | 0.8 | 0.5 | 53.7 |

*Table 3.7: Parameters set for det CT scan of the Alderson phantom*

| ***Phantom*** | ***FOV [cm]*** | ***Current [mA]*** | ***Rotation time [s]*** | ***Pitch*** | ***$CTDI_{vol}$ [mGy]*** | ***Energy [kV]*** |
|---|---|---|---|---|---|---|
| Alderson | 22 | 305 | 0.6 | 0.5 | 53.6 | 120 |

### 3.2.2 Python code

The Hounsfield Unit may differ with location in a non-circular and unsymmetrical phantom. This python code will check if this is the case for these images for three different materials in the phantom. The three different materials chosen are bone, soft tissue, and air inside the head. The Hounsfield Unit will be check for four different locations inside the Alderson phantom, for all the mono chromatic images with both types of reconstruction algorithm.

The code build on the second code, DICOM-files.py, to import the DICOM files and make a template for the 12 ROI's we are interested in. Here two different templates were made; one for the whole Alderson phantom and another for the slab Alderson phantom.

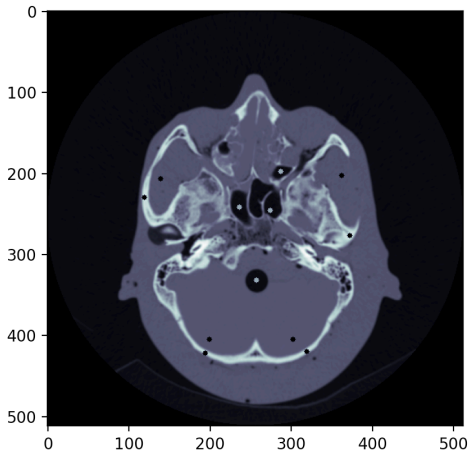


*Figure 3.6: An illustration to show how the ROIs for the different materials are located in the whole Alderson phantom. Here it shows the four ROIs in bone, soft tissue, and air inside the head.*

After importing and making a template with circular ROIs, the HU inside the ROIs is returned from the function. The HU value will only be extracted from one CT slice due to the rapid changes in anatomy in the brain from slice to slice. The ROIs have a considerable smaller size due to the width of the cranial bone. The mean HU value and the standard deviation within the ROIs was found by the code.

### 3.2.3 Treatment plan in RayStation

In order to produce proton plans the CT images were imported into RayStation treatment planning system. The monochromatic images with energy 40, 60, 80, 90, 100, 120, and 140 keV, as well as the conventional CT image with 120 kVp were imported.

To mimic a clinical relevant situation with no access to DECT scanning, a main dose plan was calculated and optimised for the 120 kVp image. Thereafter, the dose plan was applied to the monochromatic images and the difference in clinically relevant parameters was quantified. A generic CT calibration called BrillianceBig B was chosen for the 120 kVp image set. To see how the dose changed with CT calibration, two different types of CT calibration was used. The other CT calibration was called *Sandra* and was based on the HU values from the Quasar and Gammex phantom with monochromatic 80 keV reconstruction. In Appendix C both CT calibrations are tabulated.

For treatment planning a total dose of 68 Gy with 34 fractions was chosen, which corresponds to the recommendation from the DAHANCA and national program [31, 32].

#### 3.2.3.1 Delineation

Two clinical situations were simulated. Two different target volumes were delineated in the Alderson phantom by looking at two anonymised cases available in RayStation. The lesions were located in nasopharynx and pharynx, named CTV_p and CTV_n2, respectively. Figure 3.7 shows a 3D representation of the Alderson phantom with delineated areas. A set of OAR for the head and neck region are listed on the web-page for Oslo university hospital, this list was used and is tabulated in Appendix D.

Since the Alderson phantom was lacking air cavities in the nasal area, they were added by delineation and the density was set to air. In table 3.10 all the different OAR which will be of interest for this two CTV for an ear, nose and head plan is tabulated.

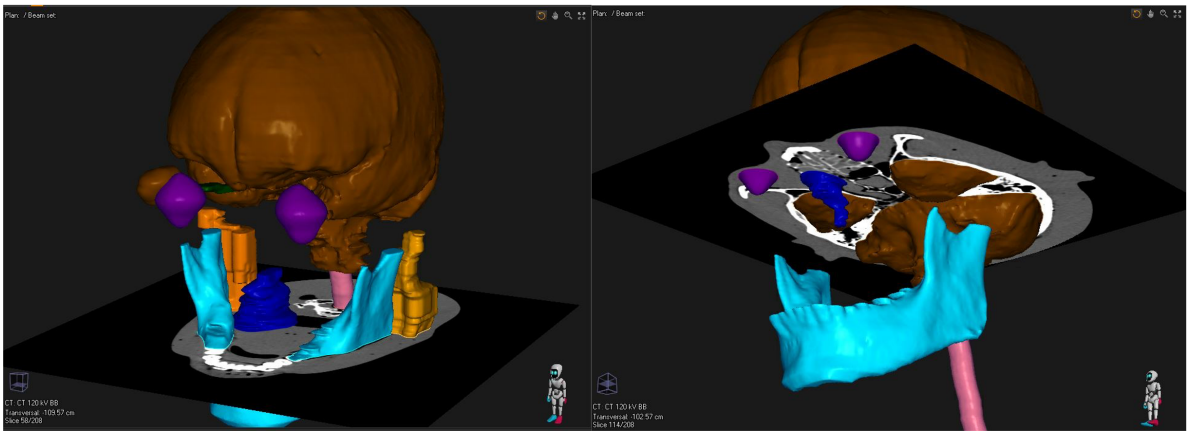

*Figure 3.7: 3D image of the Alderson phantom with delineation. The left image is of the CTV located in the pharynx area, and the image to the right is the CTV located in the nasopharynx area.*

### 3.2.3.2 Dose planning for the CTV_p

For the CTV_p, a proton plan in RayStation was already made and this setup was used. Proton modality and Pencil Beam Scanning was the treatment technique used. In table 3.8 the variables for the two beams are listed.

*Table 3.8: Beam's setup variables for the primary tumour. Range shifter or block was not used for this plan.*

| ***Beam*** | ***Isocenter*** | ***Snout name*** | ***Position*** | ***Gantry [deg]*** | ***Couch [deg]*** | ***Spot tune ID*** | ***MU/fx*** |
|---|---|---|---|---|---|---|---|
| 1 | CTV_p_1 | Snout10 | 12.58 | 250 | 0 | 3.0 | 61.01 |
| 2 | CTV_p_1 | Snout10 | 13.96 | 280 | 90 | 3.0 | 116.19 |

Before optimisation of the dose plan, a set of objectives/constraints was chosen, see table 3.9. These values were chosen from the target volumes and dose restriction, and are tabulated in Appendix D.

*Table 3.9: Constraints used for optimization of the dose plan.*

| ***Function*** | ***ROI*** | ***Description*** | ***Robust*** | ***Weight*** |
|---|---|---|---|---|
| Uniform dose | CTV | Uniform dose 68.00 Gy | * | 60.00 |
| Min dose | CTV | Min dose 63.60 Gy | * | 80.00 |
| Max dose | CTV | Max dose 71.40 Gy | * | 30.00 |
| Max dose | Body | Max dose 71.40 Gy | * | 30.00 |
| Dose fall-off | Body | [H]64.60 Gy [L] 32.00 Gy, Low dose distance 1.00 cm | | 1.00 |

| | | | |
|---|---|---|---|
| Dose fall-off | Body | [H]64.60 Gy [L] 0.00 Gy, Low dose distance 5.00 cm | 1.00 |
| Max dose | 71.4p | Max dose 70.00 Gy | 10.00 |
| Max dose | OpticNerveL | Max dose 54.00 Gy | 1.00 |
| Max dose | OpticChiasm_PVR | Max dose 60.00 Gy | 1.00 |
| Max dose | OpticChiasm | Max dose 54.00 Gy | 1.00 |
| Max dose | OpticNerveL_PVR | Max dose 60.00 Gy | 1.00 |
| Max dose | Brain | Max dose 64.00 Gy | 100.00 |
| Max dose | Eye_L | Max dose 30.00 Gy | 1.00 |
| Max DVH | Brain | Max DVH 50.00 Gy to 30% V | 1.00 |

In the *settings* an optimisation tolerance was set to be 1.000E-7 and max number of iterations was 40. To make the robust optimalisation the patient position uncertainty was set to 0.2 cm in all six directions, and the range uncertainty [%] was 3.00.

In *plan evaluation* the dose plan calculated on the 120 kVp image was recalculated on the monochromatic images, and for the two calibrations *Brialliance Big B* and *Sandra.*

#### 3.2.3.3 Dose plan on CTV_n2

For the CTV_n2, a proton plan in RayStation was already made and the same setup was used. Proton modality and Pencil Beam Scanning was the treatment technique used. In table 3.10 the variables for the three beams are listed.

*Table 3.10: Beam settings, couch rotation was set to 0, and no blocks where used.*

| ***Beam*** | ***Isocentre*** | ***Snout name*** | ***Position [cm]*** | ***Gantry [deg]*** | ***Range shifter*** | ***Spot tune ID*** | ***MU/fx*** |
|---|---|---|---|---|---|---|---|
| 1 | CTV_n_new | Snout10 | 16.63 | 230 | RS7.5cm | 3.0 | 83.89 |
| 2 | CTV_n_new | Snout10 | 14.95 | 270 | RS7.5cm | 3.0 | 81.75 |
| 3 | CTV_n_new | Snout10 | 16.18 | 310 | RS7.5cm | 3.0 | 163.89 |

A new set of objectives/constraints was chosen for the optimisation, they are tabulated in table 3.11.

*Table 3.11: Constraints used for optimisation of the dose plan.*

| ***Function*** | ***ROI*** | ***Description*** | ***Robust*** | ***Weight*** |
|---|---|---|---|---|
| Uniform dose | CTVn2 | Uniform dose 68.00 Gy | * | 60.00 |
| Min dose | CTVn2 | Min dose 65.00 Gy | * | 80.00 |
| Max dose | CTVn2 | Max dose 71.40 Gy | * | 30.00 |
| Max dose | Body | Max dose 71.40 Gy | * | 1.00 |
| Dose fall-off | Body | [H]64.60 Gy [L] 32.00 Gy, Low dose distance 1.00 cm | | 1.00 |
| Dose fall-off | Body | [H]64.60 Gy [L] 0.00 Gy, Low dose distance 5.00 cm | | 1.00 |
| Max dose | 71.4 | Max dose 70.00 Gy | | 100.00 |
| Max EUD | Parotide_R | Max EUD 24.00 Gy, Parameter A 1 | | 1.00 |
| Max EUD | SubmandGland_L | Max EUD 34.99 Gy, Parameter A 1 | | 1.00 |

All other parameters were set exactly as for the CTV_p_1 dose plan.

In *plan evaluation* the dose plan calculated on the 120 kVp image was recalculated on the monochromatic images, and for the two calibrations *Brialliance Big B* and *Sandra.*

# 4 Results

## 4.1 Single Energy CT (SECT)

To check if the CT simulation tool in python produced a photon spectrum as expected, the photon energy fluence as a function of energy $h\nu$ was plotted. The photon fluence without filtering (original), with aluminium filter, with body (phantom) filter, and both filters were plotted together to see how they differed. This is illustrated in figure 4.1.

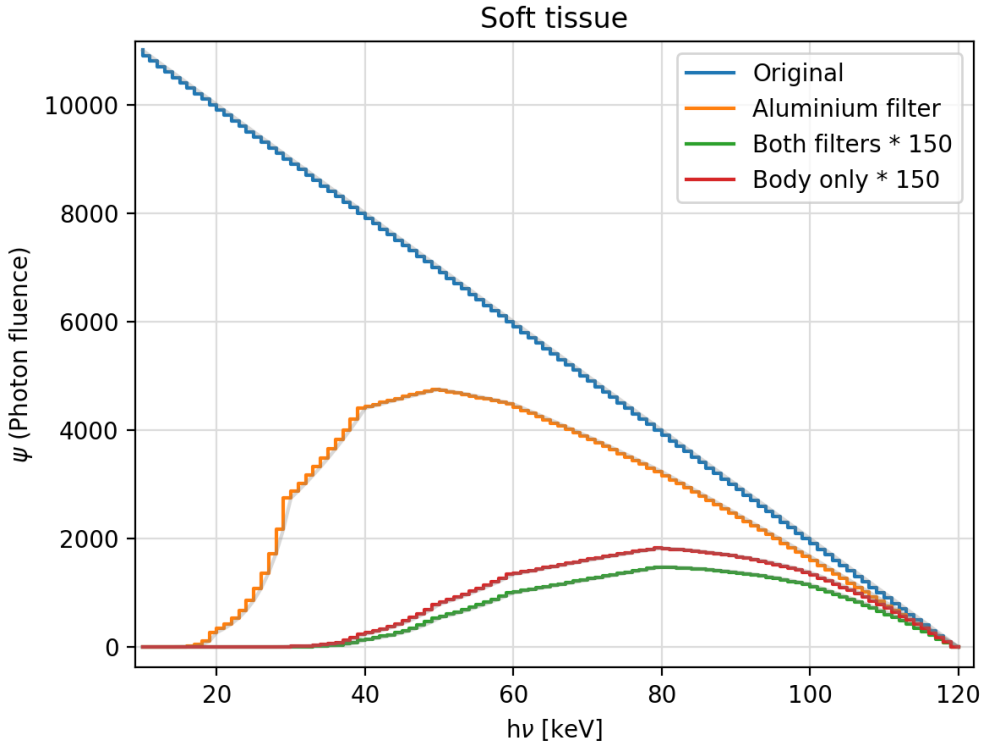


*Figure 4.1: Photon fluence as a function of energy (hν) for different filtering. The* Original *(blue line) is the original Kramer spectrum without filtering, the orange line is with aluminium filter, red line with body filtering, and green line with both aluminium and body filtering. For a better representation of how the photon fluence is attenuated through both filters and body only, they are multiplied with a constant equal to 150. The lines are stepped to illustrate that the photon energy fluence is discretizes in steps of 1 keV.*

In the result chapter the measured values will be for CT images that is reconstructed with the *Backprojection* (BP) algorithm as long nothing else is specified. To decide this the mean of the difference between HU values for BP and *iterative reconstruction* (IT) for all materials and all energies was calculated, which gave the results of 0.02 HU, which is neglectable small in a clinical setting.

SECT data was sampled from the Quasar and Gammex phantoms. Mean HU values from materials corresponding to materials in the CT simulation tool was sampled using a ROI of 10 pixels and 5 image slices. The materials were: *Inflated lung, Cortical bone (dense bone), Brain, Blood (Blood70), Polyethylene, and Adipose,* and the data are tabulated in table 4.1. When performing the simulations, it was not fully clear what representative thickness that should be employed when estimating the HU. Thus, a 120 kVp filtered spectrum for three different phantom thicknesses was made, and the least square method was used to find the thickness which minimizes the sum of squared differences between simulated and measured HU. Results are presented in table 4.1.

*Table 4.1: Measured HU values for the Quasar and Gammex phantom and estimated HU values from the simulation for six materials:* Inflated lung, Cortical bone (Dense bone), Brain, Blood (Blood70), Polyethylene, and Adipose. *In the last column the sum of squared differences between measured and simulated values are tabulated.*

| ***Material*** | Inf. Lung | Cort. bone | Brain | Blood | Polyethylene | Adipose | $\sum(HU_m - HU_s)^2$ |
|---|---|---|---|---|---|---|---|
| ***Measured HU*** | -740 | 695 | 34 | 79 | -72 | -67 | |
| ***Simulated 30cm*** | -731 | 606 | 36 | 53 | -94 | -76 | 9242 |
| ***Simulated 20 cm*** | -700 | 802 | 35 | 51 | -109 | -82 | 15492 |
| ***Simulated 3 cm*** | -744 | 1123 | 41 | 62 | -150 | -111 | 191389 |

From the result of the least square method, a thickness of 30 cm was used throughout for the CT simulation.

To see how well the HU values for measured and simulated 120 kVp correlates, both the correlation and linear regression coefficients are found, and the values are tabulated in table 4.2.

*Table 4.2: Correlation and linear regression coefficients for measured and simulated HU at 120 kVp.*

| | $r^2$ | $a \pm STD$ | $b \pm STD$ |
|---|---|---|---|
| *120 kVp* | 0.99 | $0.93 \pm 0.04$ | $-23.36 \pm 16.06$ |

The correlation between the measured mean HU values and simulated HU values is also illustrated in figure 4.2, where also an identity line with a=1 and b=0 is drawn. As seen, although the correlation is high there is some systematic deviations between the measured and simulated values.

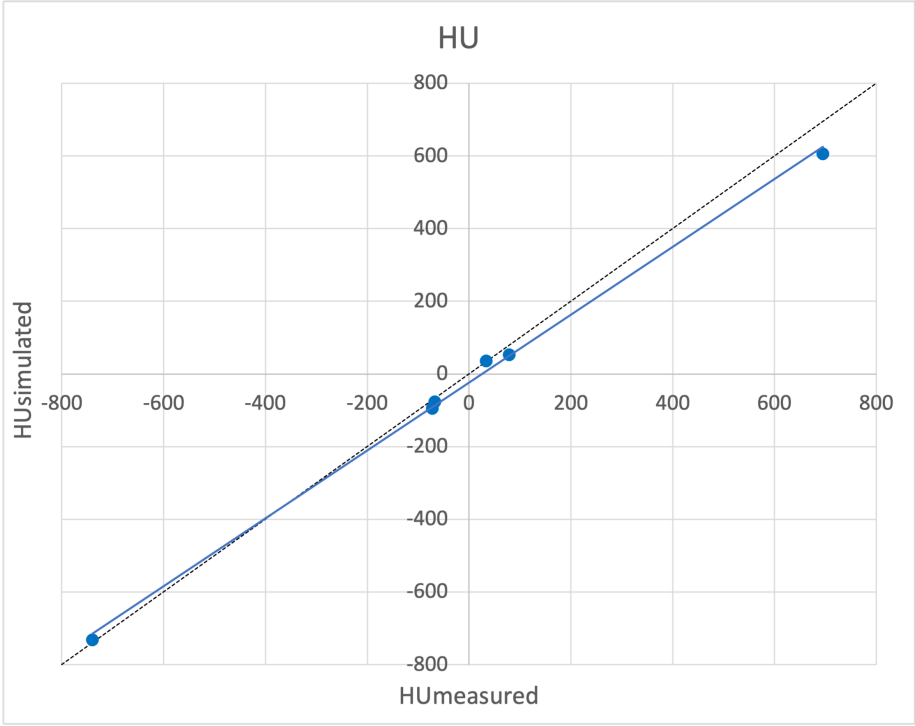


*Figure 4.2: Correlation between measured and simulated HU for 120 kVp. The blue circular points with a linear fit are the HU values, and the black stippled line is the linear fit through b=0.*

## 4.2 Dual Energy CT (DECT)

Here the simulated and measured HU values for monochromatic images derived from DECT are presented. In table 4.3 the simulated HU are tabulated, while both measured and simulated HU values for the same six materials as in chapter 4.1 are presented in figure 4.3. Six plots of how the HU values vary with energy are shown. HU values from both monochromatic and 120 kVp images are represented, and in Appendix A and B the values are tabulated.

*Table 4.3 Overview of the simulated HU values from the CT simulation code made in python. These values are calculated based on attenuation and mass density data from NIST.*

| ***Energy*** | ***Soft Tissue*** | ***Inf. Lung*** | ***Cortical Bone*** | ***Blood*** | ***Brain*** | ***Polyethylene*** | ***Adipose*** |
|---|---|---|---|---|---|---|---|
| 40 | 62 | -749 | 2522 | 47 | 73 | -211 | -152 |
| 50 | 58 | -750 | 1655 | 43 | 64 | -146 | -111 |
| 60 | 54 | -751 | 1171 | 39 | 59 | -110 | -89 |
| 70 | 53 | -751 | 960 | 38 | 57 | -95 | -80 |
| 80 | 52 | -752 | 723 | 37 | 54 | -77 | -69 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 90 | 52 | -752 | 636 | 36 | 53 | -71 | -65 |
| 100 | 51 | -752 | 543 | 36 | 53 | -63 | -61 |
| 110 | 51 | -752 | 517 | 36 | 52 | -61 | -59 |
| 120 | 51 | -752 | 489 | 36 | 52 | -59 | -58 |
| 130 | 51 | -752 | 460 | 36 | 52 | -60 | -56 |
| 140 | 51 | -752 | 429 | 35 | 51 | -55 | -55 |

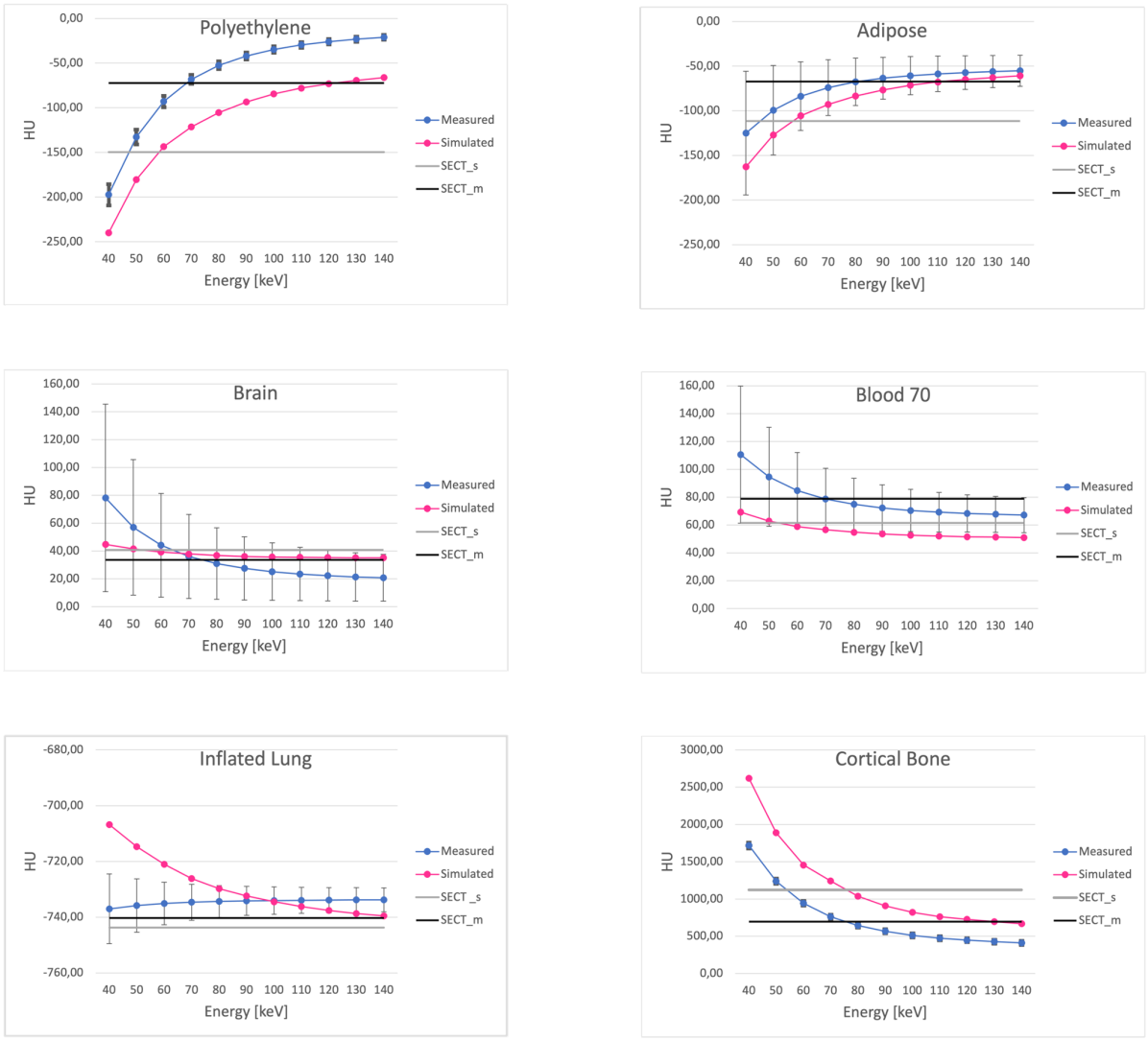


*Figure 4.3: Measured and simulated HU values versus energy from DECT monochromatic images and for 120 kVp images for the materials polyethylene, adipose, brain, blood (blood70), inflated lung, and cortical bone (dense bone). The HU value for 120 kVp is presented to give a better illustration of how HU values for the monochromatic images changes compared to kVp. The blue line (with circular points) is the measured HU value with STD. The pink line (with circular points) is the simulated HU values. Grey continuous line is the simulated monochromatic HU values, and the black continuous line is the HU values for measured 120 kVp.*

For the three materials in the Gammex phantom (*Blood, Brain, and Adipose)* a theoretical calculated HU value for monoenergetic energies is available. These values are taken from the Gammex manual and are tabulated in table 4.4.

*Table 4.4: Theoretical HU for chosen mediums taken form the Gammex phantom.*

| *HU* | *40* | *50* | *60* | *70* | *80* | *90* | *100* | *110* | *120* | *130* | *140* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Blood70 | 90 | 82 | 76 | 72 | 71 | 70 | 69 | 69 | 68 | 68 | 67 |
| Brain | 49 | 39 | 32 | 28 | 26 | 25 | 24 | 23 | 23 | 22 | 22 |
| Adipose | -140 | -107 | -90 | -80 | -73 | -69 | -66 | -65 | -63 | -62 | -62 |

To find the monochromatic image with the best fit to 120 kVp image, least squared method was used on both the HU values and the noise in the image. The three best results are tabulated in table 4.5.

*Table 4.5: Result of the least square method for $HU_{DECT}$ (keV) and $HU_{SECT}$ (kV). Only the three best results are tabulated here. The least square method for $Noise_{DECT}$ and $Noise_{SECT}$ was also calculated and tabulated here. These values give an indication of which monochromatic image is most comparable to the 120 kV image.*

| *Energy [keV]* | 70 | *80* | *90* |
|---|---|---|---|
| $\sum(HU_{DECT} - HU_{SECT})^2$ | 5908 | 3211 | 19389 |
| $\sum(Noise_{DECT} - Noise_{SECT})^2$ | 70 | 18 | 100 |

To get a better sense of how the measured and simulated data corresponds to each other, both correlation and linear regression coefficient were calculated. Linear regression for the different energies tells something about how good the data fits to a linear line $y = ax + b$, and the standard deviation for both $a$ and $b$ can be calculated. In figure 4.4 the correlation between measured and simulated HU values from monochromatic images are illustrated.

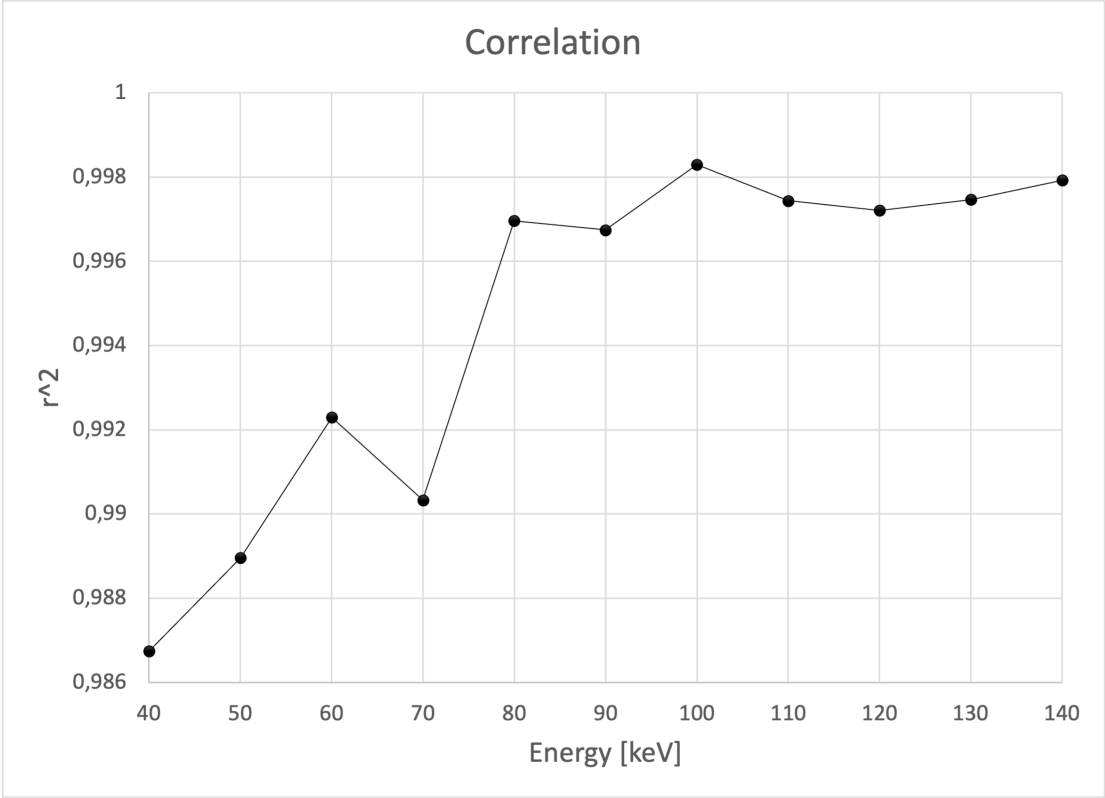


*Figure 4.4:The correlation ($r^2$) values between measured and simulated HU for the different monoenergetic energies are illustrated as black circular points on the graph.*

In table 4.6 these values are tabulated for 80 - 110 keV, which from figure 4.4 are the energies with the highest correlation.

*Table 4.6:Linear regression coefficients for 80, 90, 100, and 110 keV measured HU values. Energies chosen are the values with highest correlation between measured and simulated HU.*

| ***Energy [keV]*** | 80 | *90* | *100* | *110* |
|---|---|---|---|---|
| ***a* ± *STD*** | $1.07 \pm 0.06$ | $1.06 \pm 0.06$ | $1.04 \pm 0.04$ | $1.04 \pm 0.02$ |
| ***b* ± *STD*** | $4.74 \pm 23.64$ | $3.91 \pm 23.12$ | $-2.45 \pm 17.74$ | $-0.23 \pm 18.98$ |

As seen from earlier plots in the result, it is interesting to see how the HU value differs between the measured and the simulated energy settings. For each material the difference between the measured and simulated HU values are calculated. Results are illustrated in figure 4.5. The plot only contains information about the difference between monoenergetic values.

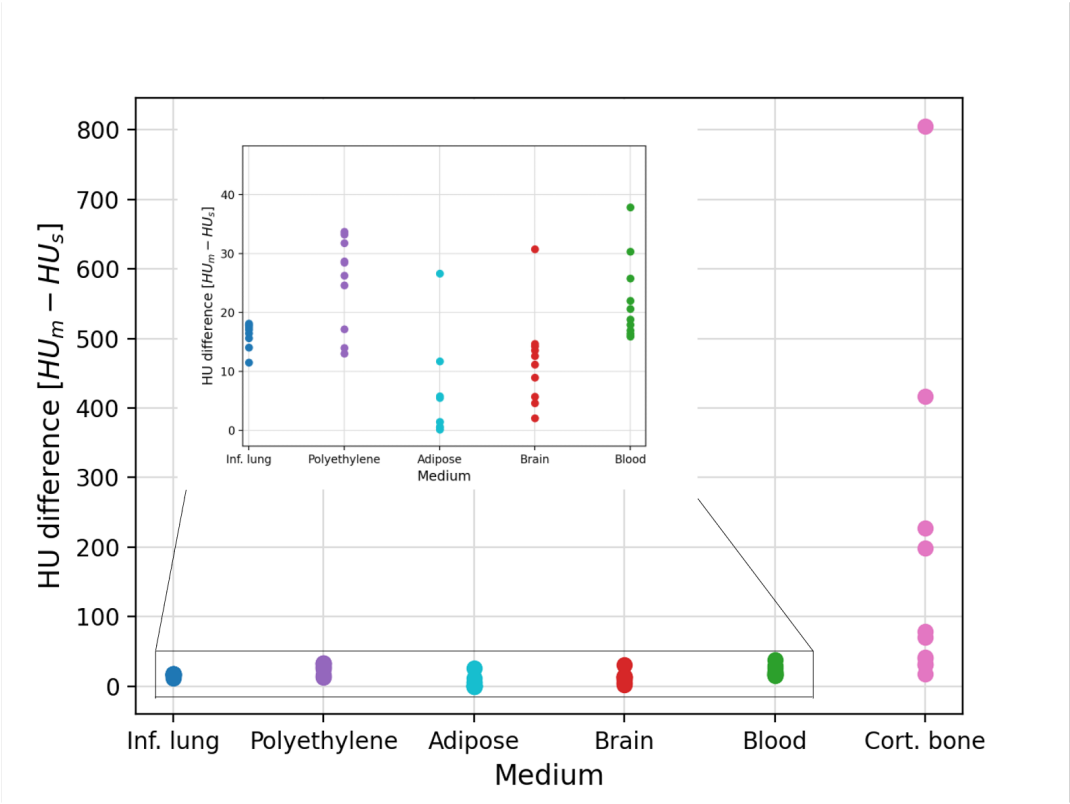


*Figure 4.5 :The difference in HU values (measured - simulated) plotted for each material for monochromatic (keV) values. The blue circular points are for inflated lung, the purple points are for polyethylene, the cyan circular points are for adipose, red circular points are for brain, green circular points are for blood, and pink circular points are for cortical bone (dense bone).*

## 4.3 Alderson

To see how HU values might change with location in a CT image, a small circular ROI was set to contain only one material a time and the mean HU was sampled. With this ROI, a local mean HU was found in four different locations for three different types of material, *soft tissue* (ST), *air* (A), and *bone* (B). The stored values can be found in Appendix F. To see how the mean HU differs between the locations, a global mean HU with STD is calculated.

In table 4.7 and 4.8 the mean HU for the four different spots called *Global mean,* and tree different material is listed, they are for the whole Alderson and slab Alderson phantom, respectively. The standard deviation of the four HU values is calculated and presented with the Global mean.

*Table 4.7: In this table values from the whole Alderson phantom is listed. The* Global mean *is the mean HU from four different locations, and the* standard aviation *of the HU is presented after the global mean. Three different materials are listed here,* Soft Tissue *(ST),* Bone *(B), and* Air *(A). The HU is measured for monoenergetic energies, and here the values from 40-140 keV with a step size of 20 is listed.*

| ***Energy*** | *40* | *60* | *80* |
|---|---|---|---|
| ***Global mean ± STD [ST]*** | $16 \pm 49$ | $6 \pm 23$ | $1.8 \pm 13.3$ |
| ***Global mean ± STD [B]*** | $3326 \pm 530$ | $1785 \pm 223$ | $1186 \pm 193$ |
| ***Global mean ± STD [A]*** | $-1028 \pm 19$ | $-999 \pm 8$ | $-987 \pm 7$ |
| **Energy** | 100 | 120 | 140 |
| ***Global mean ± STD [ST]*** | $0.3 \pm 8.9$ | $-0.6 \pm 6.6$ | $-1 \pm 5$ |
| ***Global mean ± STD [B]*** | $925 \pm 152$ | $796 \pm 132$ | $723 \pm 121$ |
| ***Global mean ± STD [A]*** | $-982 \pm 8$ | $-980 \pm 8$ | $-978 \pm 8$ |

*Table 4.8: In this table values from the slab Alderson phantom is listed. The* Global mean *is the mean HU from four different locations, and the* standard aviation *of the HU is presented after the global mean. Three different materials are listed here,* Soft Tissue *(ST),* Bone *(B), and* Air *(A). The HU is measured for monoenergetic energies, and here the values from 40-140 keV with a step size of 20 is listed.*

| ***Energy*** | *40* | *60* | *80* |
|---|---|---|---|
| ***Global mean ± STD [ST]*** | $34 \pm 4$ | $9 \pm 2$ | $-1 \pm 1$ |
| ***Global mean ± STD [B]*** | $3531 \pm 353$ | $1875 \pm 199$ | $1232 \pm 140$ |
| ***Global mean ± STD [A]*** | $-1022 \pm 52$ | $-996 \pm 6$ | $-985 \pm 13$ |
| **Energy** | 100 | 120 | 140 |
| ***Global mean ± STD [ST]*** | $-6 \pm 1$ | $-8 \pm 1$ | $-9 \pm 1$ |
| ***Global mean ± STD [B]*** | $952 \pm 114$ | $813 \pm 101$ | $735 \pm 94$ |
| ***Global mean ± STD [A]*** | $-981 \pm 21$ | $-979 \pm 24$ | $-977 \pm 27$ |

## 4.4 Image quality

Here the result of noise and CNR measurements from the Gammex and Quasar phantom will be presented. In table 4.9 the measured noise in the monochromatic images is shown for some of the materials. The noise was calculated with equation 2.10 directly as the values was sampled.

*Table 4.9: Amount of noise in both the monochromatic DECT images and 120 kVp image. Every 20 keV and materials with different HU are chosen to be showed here.*

| | *40 keV* | *60 keV* | *80 keV* | *100 keV* | *120 keV* | *140 keV* | *120 kVp* |
|---|---|---|---|---|---|---|---|
| ***Infl. Lung*** | 12.5 | 7.6 | 5.7 | 4.9 | 4.5 | 4.3 | 3.6 |
| ***Brain*** | 67.3 | 37.3 | 25.7 | 20.6 | 18.2 | 16.8 | 27.9 |
| ***Polyethylene*** | 12.0 | 6.3 | 4.3 | 3.5 | 3.1 | 2.9 | 4.2 |
| ***Dense bone*** | 16.4 | 8.8 | 5.9 | 4.7 | 4.1 | 3.8 | 6.6 |

The standard deviation of the noise is listed in table 4.10.

*Table 4.10: The change in standard deviation (STD) of the noise in the CT image of Quasar and Gammex phantom, with energy. Energies from 40 – 140 keV with step size of 20 keV is tabulated here. Also, the STD in noise for 120 kVp is listed here.*

| ***Energy*** | 40 keV | 60 keV | 80 keV | 100 keV | 120 keV | 140 keV | 120 kVp |
|---|---|---|---|---|---|---|---|
| ***STD*** | 25.9 | 14.3 | 9.8 | 7.9 | 6.9 | 6.4 | 10.5 |

The CNR was calculated from equation 2.11. CNR for chosen materials and energies are presented in table 4.11.

*Table 4.11: CNR values for chosen materials and energies.*

| | *40 keV* | *60 keV* | *80 keV* | *100 keV* | *120 keV* | *140 keV* | *120 kVp* |
|---|---|---|---|---|---|---|---|
| ***Infl. Lung*** | 18 | 29 | 39 | 45 | 49 | 51 | 84 |
| ***Brain*** | 26 | 41 | 54 | 62 | 68 | 71 | 64 |
| ***Polyethylene*** | 54 | 100 | 138 | 163 | 179 | 188 | 300 |
| ***Dense bone*** | 184 | 214 | 239 | 255 | 265 | 271 | 548 |

In figure 4.6 DECT and SECT image of the Quasar and Gammex phantom is shown. Here it is possible to see how the noise in the images changes with energy and between DECT and SECT.

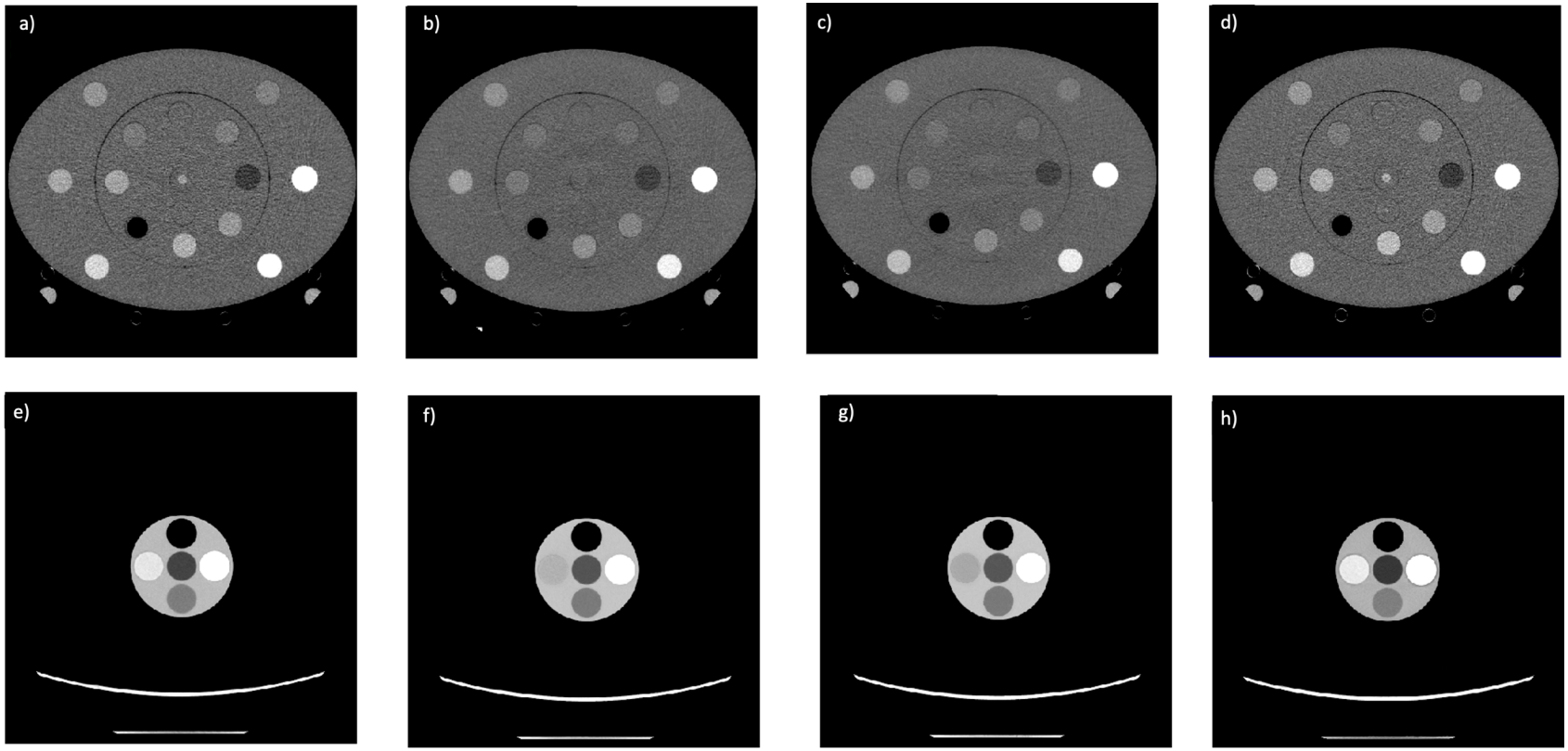


*Figure 4.6: These images show the different in noise depending on acquisition and reconstruction method. In the top row the images are of the Gammex phantom for a) 80 keV monochromatic, b) 120 keV monochromatic, c) 140 keV monochromatic, and d) 120 kVp image. On the lower row the images are of the Quasar phantom, but only the plate with the inserts, for e) 80 keV monochromatic, f) 120 keV monochromatic, g) 140 keV monochromatic, and h) 120 kVp image.*

## 4.5 SPR

The calculated SPR values are shown in table 4.12 for seven different tissues and materials. These values are used in all the figures in this sub-chapter.

*Table 4.12: SPR calculated in the python program for the mediums:* Soft tissue, Inflated lung, Adipose, Polyethylene, Blood, Brain, and Cortical bone.

| | *Soft tissue* | *Inf. Lung* | *Adipose* | *Polyethylene* | *Blood* | *Brain* | *Cortical bone* |
|---|---|---|---|---|---|---|---|
| SPR | 1.05 | 0.25 | 0.99 | 0.97 | 1.05 | 1.04 | 1.25 |

### 4.5.1 HU to SPR conversion from simulated HU values

From the simulated HU values (table 4.3) and calculated SPR values (table 4.12) a HU to SPR conversion can be plotted. Figure 4.7 shows such a plot for 120 kVp, simulating SECT, while figure 4.8 shows such a plot for 40 and 140 keV, simulating DECT, for different materials.

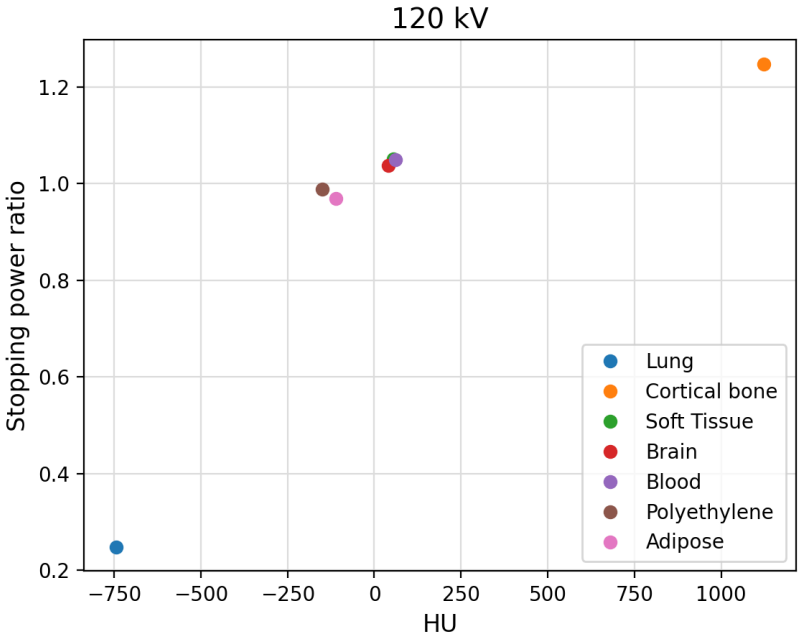


*Figure 4.7: Conversion from HU (simulated) to SPR (calculated) for different materials with 120 kVp SECT. Here the circular points represent a different material. Blue point is inflated lung, orange point is cortical bone (dense bone), green point is soft tissue (only for simulated CT), reed point is brain tissue, purple is blood, brown point is polyethylene, and the pink point is adipose.*

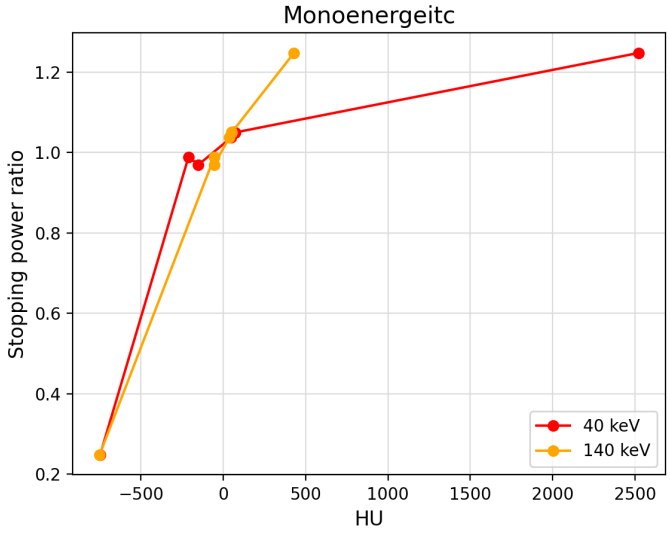


*Figure 4.8: Conversion from HU (simulated) to SPR (calculated) for the monoenergetic energies 40 keV and 140 keV, it illustrates how the HU changes with energy. The red circular points tell the HU and SPR for a CT with 40 keV, and the yellow circular points tells the HU and SPR for a CT with 140 keV. The two lines shows how a Hounsfield Unit Lookup Table (HLUT) looks like for 40 and 140 keV.*

### 4.5.2 HU to SPR conversion for measured HU values

From table 4.5 it is known that the 80 keV image results in HU values closest to the 120 kVp image. In figure 4.9 the HU to SPR conversion for the measured HU is plotted for 120 kVp, 40 keV, 80 keV, and 140 keV to illustrate the difference in a Heuristic Look Up table (HLUT).

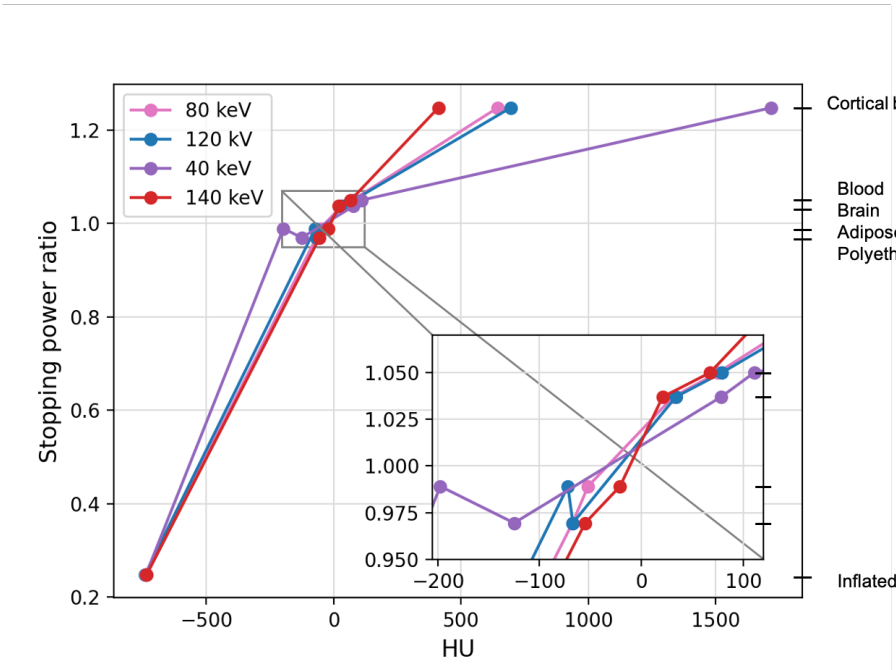


*Figure 4.9: HU to SPR conversion curve for the 120 kVp, 40 keV, 80 keV, and 140 keV. The pink line with circular points is the measured 80 keV, the blue line with circular points is the measured 120 kVp, the pruple line with circular points is the measured 40 keV, and the red line with circular points is the 120 keV. The zoomed window illustrates what happens between -200 HU and 100 HU on a larger scale.*

It seems like adipose and polyethylene have switched places for measured HU of 80 keV and 140 keV.

### 4.5.3 HU to SPR conversion for simulated vs measured values

An illustration of both the simulated and measured HU vs SPR for 120 kVp is shown in figure 4.10. From this figure it appears that the HU values are not the same but for many of the materials they are close to each other.

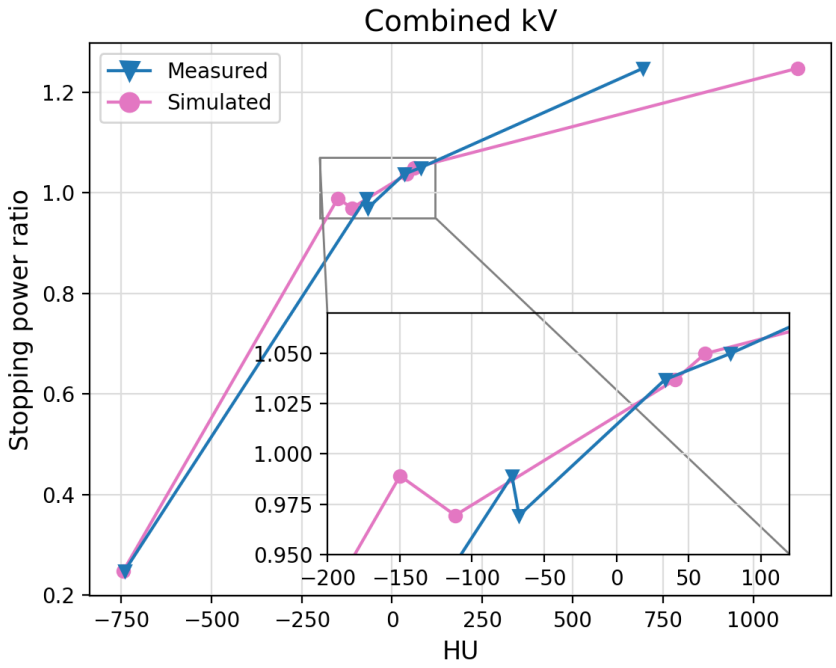


*Figure 4.10: Measured and simulated HU values at 120 kVp. The zoomed window reaches from -200 HU to 100 HU and illustrates an enlarged view of the area. The blue line with triangular points is the measured values, while the pink line with circular points is the simulated values.*

## 4.6 Dose plans

An illustration of the dose plans based on the 120 kVp CT image set for CTV_p and CTV_n2 are shown in figure 4.11 and 4.12, respectively. The objectives defined in chapter 3.2.3 were compared to different clinical goals during evaluation of the plans. These goals are tabulated in table 4.13 and 4.14 for the primary tumour and secondary tumour, respectively.

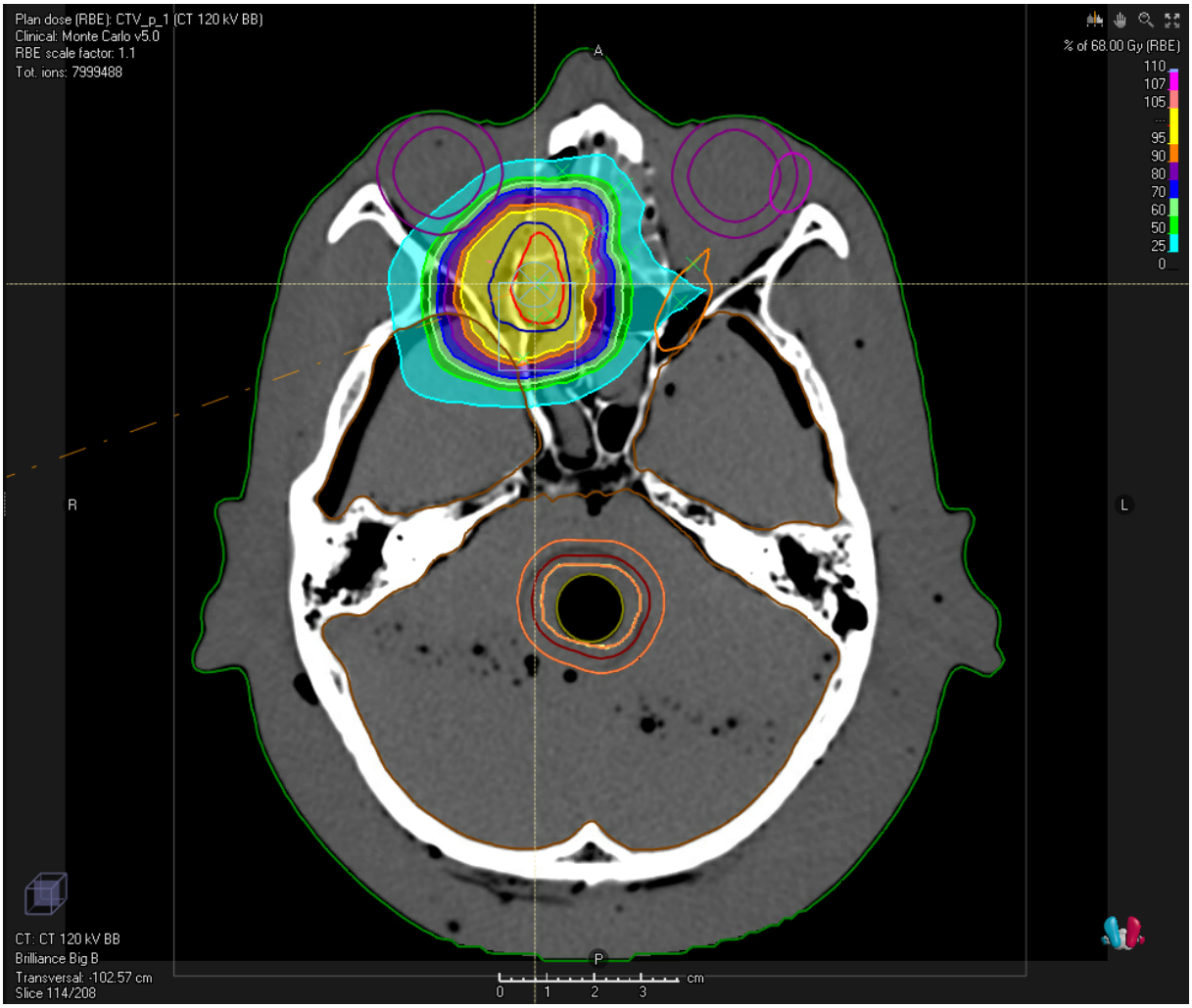


*Figure 4.11: An optimised dose plan for the tumour in the nasopharynx region. The delineation in red is the GTV and the one in blue is the CTV. The dose percentages are illustrated in colours, where the gantry angle is indicated in orange coming in from the left side of the head (the second one is from above). The colour bar in the upper right corner of the image shows the dose percentages the area gets. For a 95% coverage a yellow colour is shown. In Appendix D the colour of the different OAR is tabulated.*

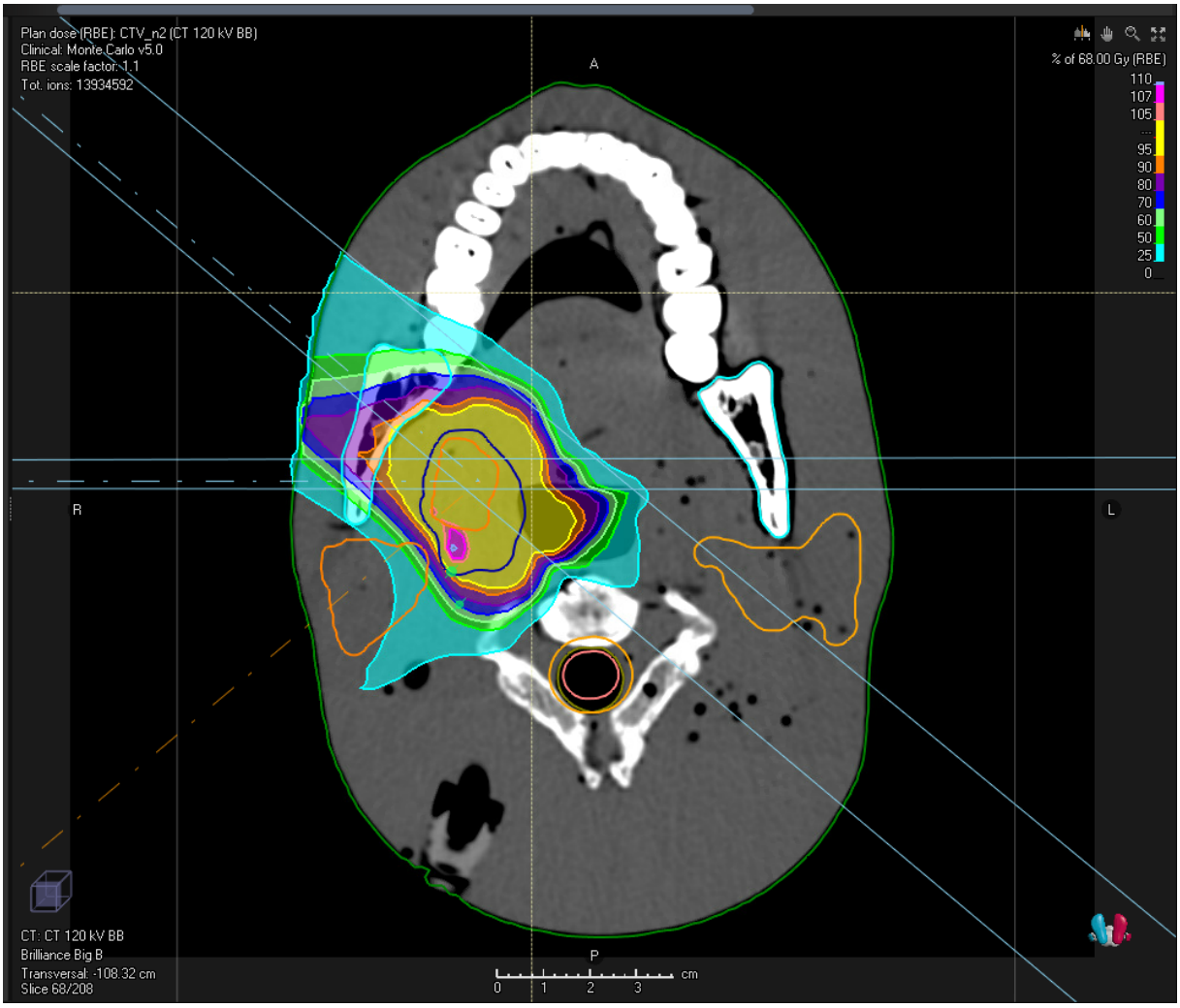


*Figure 4.12: An optimised dose plan for the tumour in the pharynx region. The delineation in orange is the GTV (inside the blue delineation) and the one in blue is the CTV. The dose percentages are illustrated in colours, where the gantry angle is indicated in orange and light blue, where one is coming in from the left side of the head (270 degree) and two with a 40 degree change from the one on the left side. The colour bar in the upper right corner of the image shows the dose percentages the area gets. For a 95% coverage a yellow colour is shown. In Appendix D the colour of the different OAR is tabulated.*

*Table 4.13: Clinical goals for the treatment plan made on the tumour in the nasopharynx region.*

| ***ROI*** | ***Clinical goal*** | ***Value*** | ***Result*** |
|---|---|---|---|
| Body | At most 70.00 Gy dose at 0.00% volume | 72.03 Gy | ! |
| Brain | At most 30% volume at 50.00 Gy dose | 0.40 % | ✓ |
| Brain | At most 60.0 Gy dose at 0% volume | 70.98 Gy | ! |
| CTV | At least 98% volume at 64.60 Gy dose | 100 % | ✓ |
| Eye_L | At most 30.00 Gy dose at 0% volume | 11.86 Gy | ✓ |
| Eye_L_PRV | At most 35.00 Gy dose at 0% volume | 16.22 Gy | ✓ |
| OpticChiasm | At most 54.00 Gy dose at 0% volume | 13.08 Gy | ✓ |
| OpticChiasm_PVR | At most 60.00 Gy dose at 0% volume | 41.59 Gy | ✓ |
| OpticNerveL | At most 54.00 Gy dose at 0% volume | 19.50 Gy | ✓ |
| OpticNerveL_PRV | At most 60.00 Gy dose at 0% volume | 24.48 Gy | ✓ |

*Table 4.14: Clinical goals for the treatment plan made on the tumour in the pharynx region.*

| ***ROI*** | ***Clinical goal*** | ***Value*** | ***Result*** |
|---|---|---|---|
| Body | At most 71.40 Gy dose at 0% volume | 76.74 Gy | ! |
| CTVn2 | At least 98% volume at 64.60 Gy dose | 99.53 % | ✓ |
| Mandible | At most 72.00 Gy dose at 0% volume | 70.07 Gy | ✓ |
| Parotide_R | At most 26.00 Gy dose at 0% volume | 12.24 Gy | ✓ |
| SpinalCord | At most 50.00 Gy dose at 0% volume | 1.18 Gy | ✓ |
| SubmandGland_R | At most 35.00 Gy dose at 0% volume | 41.48 Gy | ! |

For both plans the clinical goal for the body was not reached. The clinical goal for the brain the nasopharynx plan was not reached, while in the pharynx plan the right submandibular glad received a higher dose compared to the clinical goal.

To see how the treatment plan changed for recalculation on the chosen monochromatic images, and for a different CT calibration the D99 was sampled for the CTV and OAR. The D99 is given in percentages, and a negative percentage means that the monochromatic image or different CT calibration gets a lower dose then the original plan on the 120 kVp with CT

calibration Brilliance Big B and are tabulated in table 4.15 and 4.16 for the tumour in the nasopharynx and pharynx region, respectively.

An optimised dose plan for head and neck cancer strives for a CTV dose of 95% of total dose. This will result in tumour control, and the treatment is "good".

*Table 4.15: Change between dose coverage for the tumour in the nasopharynx region for different energies and different CT calibration, given in percentages. A negative percentage illustrates a lower recalculated dose than the original dose. The original dose was calculated on a 120 kV CT image, with BB as CT calibration. The keV settings informs which DECT image is used for recalculation of the dose. BB and Sandra is the two CT calibration used in this thesis.*

| ***Primary*** | ***BB*** | *40 keV* | *60 keV* | *80 keV* | *100 keV* | *120 keV* | *140 keV* | |
|---|---|---|---|---|---|---|---|---|
| ***CTV*** | ***D99*** | -72 | -0.4 | 0.1 | -0.1 | 0.3 | 0.2 | |
| ***Brain*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 0 | |
| ***Eye_L*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 0 | |
| ***OpticChiasm*** | ***D99*** | -7 | 14 | 14 | 21 | 7 | 21 | |
| ***OpticNerveL*** | ***D99*** | -86 | -21 | 7 | 7 | 14 | 21 | |
| | ***Sandra*** | 40 keV | 60 keV | 80 keV | 100 keV | 120 keV | 140 keV | 120 kVp |
| ***CTV*** | ***D99*** | -78 | -0.5 | -0.3 | 0 | 0.1 | 0.1 | -0.1 |
| ***Brain*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ***Eye_L*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ***OpticChiasm*** | ***D99*** | 29 | -7 | 0 | 14 | 29 | 14 | -7 |
| ***OpticNerveL*** | ***D99*** | -93 | -7 | -14 | 0 | 21 | 21 | -21 |

*Table 4.16: Change between dose coverage for the tumour in the pharynx region for different energies and different CT calibration, given in percentages. A negative percentage illustrates a lower recalculated dose than the original dose. The original dose was calculated on a 120 kV CT image, with BB as CT calibration. The keV settings informs which DECT image is used for recalculation of the dose. BB and Sandra is the two CT calibration used in this thesis.*

| | ***BB*** | *40 keV* | *60 keV* | *80 keV* | *100 keV* | *120 keV* | *140 keV* | |
|---|---|---|---|---|---|---|---|---|
| ***CTV*** | ***D99*** | -46 | -0.7 | 0.3 | 0 | 0.5 | 0.8 | |
| ***Mandible*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 0 | |
| ***Parotide_R*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 0 | |
| ***SpinalCord*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 1 | |
| ***SubmandGland_R*** | ***D99*** | 0.2 | 0 | 0 | 0 | 0 | 0 | |
| | ***Sandra*** | 40 keV | 60 keV | 80 keV | 100 keV | 120 keV | 140 keV | 120 kVp |
| ***CTV*** | ***D99*** | -46 | -0.8 | 0.1 | 0.3 | 0.5 | 0.7 | 0.1 |
| ***Mandible*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ***Parotide_R*** | ***D99*** | -0.1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ***SpinalCord*** | ***D99*** | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| ***SubmandGland_R*** | ***D99*** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# 5 Discussion

## 5.1 SECT

In *Dual energy CT in Oncology* (2015) [14] , a typical x-ray spectrum used in medical CT obtained after standard pre-filtration is illustrate. Comparing that graph with the simulated one after aluminium filtering in figure 4.1 a similar shape is seen. In *Dual energy CT in Oncology* the graph also has two peaks for a certain energy due to characteristic x-ray from the anode. The target in an x-ray tube is often tungsten, which has an atomic number of $Z = 74$ so characteristic x-rays around 58 keV and 59 keV would be expected. The presence of these peaks might affect the simulated HU values for a 120 kVp CT simulation. Still, characteristic x-rays will contribute with only a few percent of the total energy fluence, and was not included in the simulated x-ray beam due to simplifications made in the program.

The method of least squares was used to make a conclusion about what thickness the material in the CT simulation should have. A thickness of 30 cm gave simulated HU data for 120 kVp with best fit to the measured HU. 30 cm thickness is the same as the thickness of the Gammex phantom, so it is rather consistent. In a CT scanner the photon beam traverses through the whole phantom, not only the inserts, so it makes sense that the simulated photon beam also has to traverse the same distance to result in equal data. Another possibility would be to make the photon beam traverse a material that is the same as the phantom it is made of, before guiding it through the medium we are interested in and then evaluate the HU values.

From figure 4.2 we can see that the correlation between 120 kVp measured and simulated is quite good; about 0.99. But the linear regression tells us that two regression coefficients $a$ and $b$ are significantly different from unity and zero, respectively. This means that even though the data are highly correlated, there a systematic difference. This was not totally unexpected, due to the many simplifications in making the x-ray spectrum in the CT simulation tool in python. All in all, the simulation is a good approximation of simplified CT scanner, and has helped us in understanding the impact of x-ray spectrum attenuation on HU value.

## 5.2 DECT

For the monoenergetic energies HU was both measured and simulated. From theory we know that attenuation varies with energy due to energy-dependent photon interactions. For lower energies the photoelectric effect is the main interaction, but for higher energies the Compton scattering is the main interaction. Unlike the photoelectric effect, the probability of Compton scattering depends on the electron density and not on the atomic number of the scattering medium. The differences in electron density between different material are small. One would assume that for a material with a high atomic number, the change in HU with x-ray energy would be larger than for material with a low atomic number. This is because the photoelectric

effect is strongly dependent on atomic number *and* photon energy. Also *Dual energy CT in Oncology [14]* has shown this dependency for soft tissue and iodine.

In figure 4.3 one can see that the HU value varies much more for cortical bone, which has a high atomic number, compared with all the other materials. For a material with an attenuation coefficient close to, or smaller than, that of water, equation 2.9 would lead us to the conclusion that we would see the opposite effect compared to materials with high atomic values. The decrease in HU for a given material compared to that in water would lead to a higher attenuation for higher energies, and we would see an increase in HU with energy. This is seen in figure 4.3 for polyethylene and adipose. Also, both polyethylene and adipose has a lower density than water.

**Best fit to of measured data**

From the vendor of the DECT scanner it is informed that a reconstructed monochromatic image of 74 keV is expected to give the same results as a 120 kVp CT image. The least squares method was used to identify which monochromatic image series from DECT fits best to the 120 kVp SECT in terms of HU values. The numbers in table 4.5 indicates that 80 keV monochromatic image has the smallest least square difference between HU and noise, and this gives that the two images are most similar. This is not as the vendor has expected, but is in line with *Wohlfhart et al.* [5]. From a similar setup as in this work they found that monochromatic image of 79 keV gives the best results. In the same publication they present measured HU values for various inserts with tissue equivalent material from SECT 120 kVp and DECT monochromatic 79 keV images. For the inserts Bone with 10% $CaCO_3$ ($\rho = 1.17\ g/cm^3$) and Bone with 30% $CaCO_3$ ( $\rho = 1.34\ g/cm^3$) they found a difference of 11 and 46 HU, respectively, between the two image sets. These figures correspond well to the difference seen in the present work for the inserts Inner bone ($\rho = 1.12\ g/cm^3$) and Dense bone ($\rho = 1.41\ g/cm^3$). However, the measured HU values in *Wohlfhart et al* are all over a bit higher for corresponding insert (Brain and Adipose) compared to the measurement shown in figure 4.3. This illustrates the uncertainties in using HU for SPR calculation.

**Correlation between simulated and measured DECT**

A correlation between the simulated and measured monoenergetic energies gives an impression of how different the values are, but at the same time a linear regression would give more information about how the data are related to each other. The correlation was illustrated in figure 4.4 and the was between 0.986 and 0.99. The *b* value in the linear regression was not significantly different from 0 for any of the energies. For the energies above 60 keV also the *a* value in the linear regression was not significantly different from 1. This implies that CT simulation program were able to simulate monochromatic HU better than a range of energies with 120 kVp. It seems like the CT simulation works better for a monoenergetic beam than a regular one, which may be expected as the simulated Kramer spectrum is an approximation.

We have seen that the HU values for measured and simulated monochromatic images changes with energy. In figure 4.5 this change is well illustrated. It shows that for cortical bone (who has a high atomic number) the error between measured and simulated is much greater than for the other type of media used in this thesis. It also shows that the differences for adipose and inflated lung are the smallest.

## 5.3 Alderson

A variation in HU for different positions in the Alderson phantom is as expected; the HU number will be affected by the surroundings of the ROI. In table 4.7 and 4.8 the mean HU (global HU) over the four locations is tabulated with the standard deviation. The Alderson phantom used in this thesis is also of significant age and is made of real bones in the head. One would thus expect a gradual degrading of the skeleton and hence a larger variation in HU measurements. Moreover, due to the small and sometimes patchy bone areas in the head of the phantom a small ROI had to be chose, so the mean HU calculated will not be from a large stack of numbers and an uncertainty would rise.

## 5.4 Image Quality

In [33], an optimal energy of $78.5 \pm 5.0$ keV was reported, where the result of image noise and CNR calculation was best. Also, [5] found a low image noise at 79 keV MonoCT derived from 80/140 kVp DECT scans. In this study the amount of noise seems to descend with energy for the monochromatic images, as seen in table 4.9

From the least square result for measured HU, for both monochromatic and 120 kVp images, 80 keV monochromatic reconstruction image has noise values closest to the 120 kVp. When we look at the amount of noise in table 4.5 it shows that the 80 keV monochromatic image has a lower noise for most of the tissues. This can impact the HLUT, if the noise is large.

In table 4.10 the standard deviation in the image noise is tabulated. These numbers show that the error in the noise is minimized for higher energies. This is also seen for the CNR wich is tabulated in table 4.11. In [33], it was found that the image noise was decreasing until around 80 keV before it increased a bit and got stabilized. This was also the result for CNR calculation. The reason for this different result might be due to the algorithms used for creating the monochromatic images, or what type of DECT scanner is used. These results indicates that the CNR is dependent on the noise, so image with lowest noise may determine the maximum CNR in the monochromatic images.

In table 4.9 we also see that the Gammex phantom gives rise to more noise. The reason for this might be that the insert goes through the whole Gammex phantom, while in the Quasar phantom a plate with lower hight and width, is placed on the front of the phantom with the

inserts. And since we have set a dose and a current to the whole phantom, the smaller plate in the Quasar phantom will get the same photon fluence (and dose) as the rest of the phantom and a higher dose. Also, the amount of noise is highly dependent on scan modality and reconstruction parameters.

## 5.5 SPR

The SPR was calculated in the python program for seven different materials. In figure 4.9 a conversion of HU to SPR (HLUT) for both 40 keV, 80 keV, 140 keV and 120 kVp are illustrated together. This image illustrates the difference in the corresponding HLUTs. The 80 keV and 120 kVp graphs are close in value for adipose and polyethylene have made the line between them straighter. This could be a better approach, as this would imply that interpolations between these HUs are more accurate. But still the graphs in 4.9 are quite similar, which mean we would expect a similar dose distribution in proton therapy planning.

Unfortunately, it is difficult to compare figure 4.9 with the corresponding figure in *Wohlfhart et al* [5], since the various tissue inserts are not easily identified. However, it seems like the curves around the cortical bone for the SPR values correspond. *Wholfhart et al* included bone tissue material with much higher mass density (Bone with 50% $CaCO_3$, $\rho = 1.56\ g/cm^3$) and showed that the two curves deviated more for such material.

## 5.6 Proton treatment planning

A CTV located in the nasopharynx was planned with two beams. From table 4.13 it is seen that 100% of the CTV has received a dose of 64.60 Gy (95% of prescribed dose) or higher. This is also seen in figure 4.11 where the 95% isodose fully cover the CTV. The clinical goal that 0% of the whole body should receive at most 70 Gy was exceeded with 3%, which is considered not to have any clinical consequence. Also, the max dose to the brain is exceeded with 18 %. But we can see that the brain gets a 50.0 Gy dose to 0.4 % of the volume, which indicates that the max dose to the brain is in a very small area. In this treatment plan the right eye is intentionally not taken into consideration since the probability of saving it is small due to the location of the tumour.

A CTV located in the pharynx was used to make a second proton therapy plan. Here as well it seems like the tumour gets a 95 % dose coverage of whole tumour, by looking into table 4.14 it shows that the CTV got a 99.5% dose coverage with a 95 % dose. One of the two goals that were not reached was the maximum dose to the body, since it was exceeded with 8 %. The

clinical goal that 0% of the right submandibular gland should at most receive 70 Gy was also exceeded by 6.5 Gy.

From these data we would say that both treatment plans have sufficient quality. The main aim with the treatment plan was to see how the dose distribution will change when the plans are copied to monochromatic images from a DECT scanner. Also, it was interesting to see how a change in CT calibration affected the dose calculations. To quantify the changes the D99 for the CTV and OAR were evaluated. In table 4.15 the variation of these parameters is tabulated, and the only monochromatic image that stands out is the 40 keV images. For this image a reduction in D99 was around 70 % for a CT calibration that is the same as the one used for 120 kVp. When a CT calibration based on the HU values from the monochromatic 80 keV image is chosen, the dose to the CTV in D99 was around 75 %. Since the change in HU value, image noise, and CNR from a 120 kVp image to a monochromatic 40 keV image is quite large, this result was expected. And in a clinically perspective monochromatic 40 keV images will not be used for treatment planning.

Implementation of monochromatic images clinically is an intermediated step towards making DECT the preferred scanner for use in radiotherapy. The goal with the monochromatic image is to get a DECT image who will give at least the same uncertainties a 120 kVp CT images has today with the use of a HLUT. Important factors in a clinically perspective to the HU is noise and CNR. With a lower noise the error in the HLUT will be lower, and with an increased CNR delineation of tumours as well as OAR might become easier to provide.

Next step would be to receive information about the electron density and Stopping power number directly from the DECT image. For this the monochromatic image might not be of best use,but is to be further investigated. Ultimate goal is to directly calculate

# 6 Conclusion

We have gained an understanding of how uncertainties in HU will impact the HU-SPR conversion, i.e., the HLUT, which is of importance in dose calculations for proton therapy planning. A DECT scanner can contribute to monochromatic images. The monochromatic images illustrated the impact of CT x-ray energy on HU value, and we found a best approximation to a conventional 120 kVp SECT image with 80 keV monochromatic DECT image. It was seen that the 80 keV resulted in a straighter line in the HLUT, which will give more precise SPR values.

The result of the proton treatment planning did not show any significant changes for the majority of the monochromatic images.

Implementation of monochromatic images clinically is an intermediated step towards making DECT the preferred scanner for use in radiotherapy. The goal with the monochromatic image is to get a DECT image who will give at least the same uncertainties a 120 kVp CT images has today with the use of a HLUT. Important factors in a clinically perspective to the HU is noise and CNR. With a lower noise the error in the HLUT will be lower, and with an increased CNR delineation of tumours as well as OAR might become easier to provide.

The ultimate goal would be to receive information about the electron density and Stopping power number directly from the DECT image. For this the monochromatic image might not be of best use, but is to be further investigated.

Due to time limitations the mass density used in this thesis for SPR calculation was from NIST [30], a further investigation of SPR calculation for mass density corresponding to the material is a possibility.

# References


1. Olsen, D.R., Ø. Bruland, G. Frykholm & I. Norderhaug, *Protonterapi*. 2006, Nasjonalt kunnskapssenter for helsetjenesten: FHI. p. 63.
2. Hall, E.J. & A.J. Giaccia, *Radiobiology for the radiologist*. Eighth, international edition. ed. 2019, Philadelphia: Wolters Kluwer.
3. Institute, N.C. *Radiation Therapy to Treat Cancer*. 2019 08.01.2019 [cited 2019 08.01]; Available from: https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy.
4. Dale, E. & E. Waldeland, *Protonterapi - en realitet i Norge fra 2023.* 2018.
5. Wohlfahrt, P., et al., *Clinical Implementation of Dual-energy CT for Proton Treatment Planning on Pseudo-monoenergetic CT scans.* Int J Radiat Oncol Biol Phys, 2017. **97**(2): p. 427-434.
6. Hudobivnik, N., et al., *Comparison of proton therapy treatment planning for head tumors with a pencil beam algorithm on dual and single energy CT images.* Med Phys, 2016. **43**(1): p. 495.
7. Attix, F.H., *Introduction to Radiological Physics and Radiation Dosimetry*. 1. Aufl. ed. 2008: Wiley-VCH.
8. Schneider, U., E. Pedroni & A. Lomax, *The calibration of CT Hounsfield units for radiotherapy treatment planning.* Phys Med Biol, 1996. **41**(1): p. 111-24.
9. Mayles, P., A.E. Nahum & J.-C. Rosenwald, *Handbook of radiotherapy physics : theory and practice*. 2007, Taylor & Francis: New York.
10. Kalender, W., *Computed tomography : fundamentals, system technology, image quality, applications*. 3rd rev. ed. ed. 2011, Erlangen: Publicis.
11. Flower, M.A. & S. Webb, *Webb's physics of medical imaging*. 2nd ed. ed. Series in medical physics and biomedical engineering. 2012, Boca Raton, Fla: CRC Press.
12. Jensen, K., *Iterative reconstruction in CT imaging; Image quality and radiation doses*, in *Department of physics*. 2020, University of Oslo. p. 117.
13. Heismann, B., B. Schmidt & T. Flohr, *Spectral Computed Tomography*. 2012, Bellingham, Washington USA: SPIE. 119.
14. De Cecco, C.N., A. Laghi, U.J. Schoepf & F.G. Meinel, *Dual Energy CT in Oncology*. 2015, Springer International Publishing : Imprint: Springer: Cham.
15. Yeung, D. & J. Palta, *Proton Therapy*, in *Encyclopedia of Radiation Oncology*, L.W. Brady & T.E. Yaeger, Editors. 2013, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 675-690.
16. Lee, N.Y., et al., *Target Volume Delineation and Treatment Planning for Particle Therapy : A Practical Guide*. 2018, Springer International Publishing : Imprint: Springer: Cham.
17. Tsuboi, K., T. Sakae & A. Gerelchuluun, *Proton Beam Radiotherapy : Physics and Biology*. 2020, Springer Singapore : Imprint: Springer: Singapore.
18. Grant, J.D. & J.Y. Chang, *Proton-Based Stereotactic Ablative Radiotherapy in Early-Stage Non-Small-Cell Lung Cancer.* BioMed Research International, 2014. **2014**: p. 389048.
19. Maarten Schippers, J. *Beam Transport Systems for Particle Therapy*. 2018. arXiv:1804.08551.
20. Raysearch, *RayStation 10B Reference Manual*. 2020: Stockholm. p. 370.
21. ICRU, *ICRU Report 78*. 2007: Washington.

22. Aarberg, A.E., *Proton therapy of head and neck cancer: evaluation of PTV-based and robust optimized IMPT versus VMAT*, in *Physics*. 2017, Norwegian University of Scinece and Technology. p. 72.
23. Unkelbach, J. & H. Paganetti, *Robust Proton Treatment Planning: Physical and Biological Optimization.* Semin Radiat Oncol, 2018. **28**(2): p. 88-96.
24. ICRU, *ICRU Report 50*. 1993: Washington.
25. ICRU, *ICRU Report 62*. 1999: Washington.
26. QA, M. *Adaptable Dosimetric and Nondosimetric QA*. 2020 [cited 2020 09]; Available from: https://modusqa.com/wp-content/uploads/2020/09/Modus-QA-Product-Data-Sheet-MP-Body.pdf.
27. Corporation, S.N. *Advanced Electron Density Phantom*. 2021 [cited 2021 08.06]; Available from: https://www.sunnuclear.com/products/advanced-electron-density-phantom.
28. Espe, I.K., *Kvalitetskontroll av ikke-dosimetriske parametre ved CT-basert planlegging av stråleterapi.* 2006, Statens Strålevern: Østerås. p. 33.
29. Corporation, S.N., *User's Guide, Multi-Energy CT Phantom*. 2017, Gammex Inc.: Middleton. p. 16.
30. Hubbell, J.H. & S.M. Seltzer. *Tables of X-Ray Mass Attenuation Coefficients and Mass Energy-Absorption Coefficients from 1 keV to 20 MeV for Elements Z=1 to 92 and 48 Additional Stubstances of Dosimetric Interest*. 1989 11.12.2019 [cited 2009 17.09].
31. DAHANCA. 1976; The national database for head and neck cancer ]. Available from: https://www.dahanca.dk/IndexPage.
32. Helsedirektoratet, *Hode-hals-kreft - handlingsprogram [nettdokument]*. 2020, Helsedirektoratet: Oslo. p. 202.
33. Wang, T., et al., *Optimal virtual monoenergetic image in "TwinBeam" dual-energy CT for organs-at-risk delineation based on contrast-noise-ratio in head-and-neck radiotherapy.* J Appl Clin Med Phys, 2019. **20**(2): p. 121-128.

# *Appendices*

# *Appendix A*

Measured values Gammex and Quasar phantoms

## A 1 HU

*Table A.1: HU from the monochromatic energies with the DECT scan, the algorithm used for reconstruction is backprojection. The values are from the Gammex phantom.*

| ***Energy [keV]*** | ***Blood70*** | ***Blood40*** | ***Brain*** | ***Water*** | ***Blood*** | ***Adipose*** |
|---|---|---|---|---|---|---|
| 40 | 111 | 77 | 78 | 6 | 148 | -125 |
| 50 | 95 | 62 | 57 | 4 | 129 | -99 |
| 60 | 85 | 53 | 44 | 3 | 117 | -84 |
| 70 | 79 | 47 | 36 | 2 | 110 | -74 |
| 80 | 75 | 43 | 31 | 1 | 105 | -68 |
| 90 | 72 | 41 | 28 | 1 | 102 | -64 |
| 100 | 70 | 39 | 25 | 1 | 100 | -61 |
| 110 | 69 | 38 | 23 | 0 | 98 | -59 |
| 120 | 68 | 37 | 22 | 0 | 97 | -57 |
| 130 | 68 | 36 | 21 | 0 | 97 | -56 |
| 140 | 67 | 36 | 21 | 0 | 96 | -55 |

*Table A.2: HU from the monochromatic energies with the DECT scan, the algorithm used for reconstruction is iterative reconstruction. The values are from the Gammex phantom.*

| ***Energy [keV]*** | ***Blood70*** | ***Blood40*** | ***Brain*** | ***Water*** | ***Blood*** | ***Adipose*** |
|---|---|---|---|---|---|---|
| 40 | 111 | 77 | 78 | 6 | 148 | -125 |
| 50 | 95 | 62 | 57 | 4 | 129 | -100 |
| 60 | 85 | 53 | 44 | 3 | 117 | -84 |
| 70 | 79 | 47 | 36 | 2 | 110 | -74 |
| 80 | 75 | 43 | 31 | 1 | 105 | -68 |
| 90 | 72 | 41 | 28 | 1 | 102 | -64 |
| 100 | 71 | 39 | 25 | 0 | 100 | -61 |
| 110 | 69 | 38 | 23 | 0 | 98 | -59 |
| 120 | 68 | 37 | 22 | 0 | 97 | -57 |
| 130 | 68 | 36 | 21 | 0 | 96 | -56 |
| 140 | 67 | 36 | 21 | 0 | 96 | -55 |

*Table A.3: HU from the monoenergetic energies with the DECT scan, the algorithm used for reconstruction is backprojection. The values are from the Quasar phantom.*

| ***Energy [keV]*** | ***Inflated lung*** | ***Dense bone*** | ***Water*** | ***Inner bone*** | ***Polyethylene*** |
|---|---|---|---|---|---|
| 40 | -737 | 1717 | 60 | 640 | -197 |
| 50 | -736 | 1238 | 50 | 444 | -133 |
| 60 | -735 | 944 | 44 | 324 | -93 |
| 70 | -735 | 761 | 40 | 249 | -68 |
| 80 | -734 | 643 | 37 | 201 | -52 |
| 90 | -734 | 566 | 36 | 169 | -42 |
| 100 | -734 | 513 | 35 | 147 | -35 |
| 110 | -734 | 474 | 34 | 132 | -30 |
| 120 | -734 | 448 | 33 | 121 | -26 |
| 130 | -734 | 427 | 33 | 112 | -23 |
| 140 | -734 | 411 | 32 | 106 | -21 |

*Table A.4: HU from the monoenergetic energies with the DECT scan, the algorithm used for reconstruction is iterative reconstruction. The values are from the Quasar phantom.*

| ***Energy [keV]*** | ***Inflated lung*** | ***Dense bone*** | ***Water*** | ***Inner bone*** | ***Polyethylene*** |
|---|---|---|---|---|---|
| 40 | -737 | 1717 | 60 | 641 | -197 |
| 50 | -736 | 1238 | 50 | 444 | -133 |
| 60 | -735 | 943 | 44 | 324 | -93 |
| 70 | -735 | 761 | 40 | 249 | -68 |
| 80 | -734 | 643 | 37 | 201 | -52 |
| 90 | -734 | 566 | 36 | 169 | -42 |
| 100 | -734 | 512 | 35 | 147 | -35 |
| 110 | -734 | 474 | 34 | 132 | -30 |
| 120 | -734 | 448 | 33 | 121 | -26 |
| 130 | -734 | 427 | 33 | 112 | -23 |
| 140 | -734 | 411 | 32 | 106 | -21 |

# A 2 Noise

*Table A.5: Noise from the monoenergetic energies with the DECT scan, the algorithm used for reconstruction is backprojection. The values are from the Gammex phantom.*

| ***Energy [keV]*** | ***Blood70*** | ***Blood40*** | ***Brain*** | ***Water*** | ***Blood*** | ***Adipose*** |
|---|---|---|---|---|---|---|
| 40 | 49.2 | 55.5 | 67.3 | 61.7 | 67.8 | 69.3 |
| 50 | 35.6 | 40.1 | 48.7 | 44.3 | 48.9 | 50.2 |
| 60 | 27.3 | 30.6 | 37.3 | 33.6 | 37.3 | 38.5 |
| 70 | 22.1 | 24.8 | 30.2 | 27.1 | 30.2 | 31.2 |
| 80 | 18.8 | 21.0 | 25.7 | 23.0 | 25.6 | 26.6 |
| 90 | 16.7 | 18.6 | 22.7 | 20.4 | 22.7 | 23.5 |
| 100 | 15.2 | 16.9 | 20.6 | 18.5 | 20.6 | 21.4 |
| 110 | 14.2 | 15.7 | 19.2 | 17.3 | 19.2 | 19.9 |
| 120 | 13.5 | 14.9 | 18.2 | 16.4 | 18.2 | 18.8 |
| 130 | 12.9 | 14.2 | 17.4 | 15.8 | 17.4 | 18.1 |
| 140 | 12.5 | 13.8 | 16.8 | 15.3 | 16.8 | 17.4 |

*Table A.6: Noise from the monoenergetic energies with the DECT scan, the algorithm used for reconstruction is iterative reconstruction. The values are from the Gammex phantom.*

| ***Energy [keV]*** | ***Blood70*** | ***Blood40*** | ***Brain*** | ***Water*** | ***Blood*** | ***Adipose*** |
|---|---|---|---|---|---|---|
| 40 | 30.0 | 34.6 | 40.9 | 39.2 | 42.0 | 42.0 |
| 50 | 21.6 | 24.8 | 29.5 | 27.9 | 30.1 | 30.5 |
| 60 | 16.5 | 18.9 | 22.5 | 21.1 | 22.9 | 23.4 |
| 70 | 13.4 | 15.3 | 18.3 | 17.0 | 18.4 | 19.0 |
| 80 | 11.4 | 13.0 | 15.5 | 14.4 | 15.6 | 16.2 |
| 90 | 10.1 | 11.4 | 13.7 | 12.8 | 13.8 | 14.4 |
| 100 | 9.3 | 10.4 | 12.5 | 11.7 | 12.5 | 13.1 |
| 110 | 8.7 | 9.7 | 11.6 | 11.0 | 11.6 | 12.2 |
| 120 | 8.3 | 9.2 | 11.0 | 10.5 | 11.0 | 11.6 |
| 130 | 8.0 | 8.8 | 10.5 | 10.1 | 10.6 | 11.1 |
| 140 | 7.7 | 8.5 | 10.2 | 9.8 | 10.2 | 10.7 |

*Table A.7: STD from the monoenergetic energies with the DECT scan, the algorithm used for reconstruction is backprojection. The values are from the Quasar phantom.*

| ***Energy [keV]*** | ***Inflated lung*** | ***Dense bone*** | ***Water*** | ***Inner bone*** | ***Polyethylene*** |
|---|---|---|---|---|---|
| 40 | 12.5 | 16.3 | 12.5 | 14.5 | 12.0 |
| 50 | 9.6 | 11.7 | 8.9 | 10.4 | 8.4 |
| 60 | 7.6 | 8.8 | 6.8 | 7.9 | 6.3 |
| 70 | 6.5 | 7.1 | 5.5 | 6.4 | 5.1 |
| 80 | 5.7 | 5.9 | 4.7 | 5.4 | 4.3 |
| 90 | 5.2 | 5.2 | 4.1 | 4.8 | 3.8 |
| 100 | 4.9 | 4.7 | 3.7 | 4.3 | 3.5 |
| 110 | 4.7 | 4.4 | 3.5 | 4.0 | 3.2 |
| 120 | 4.5 | 4.1 | 3.3 | 3.8 | 3.1 |
| 130 | 4.4 | 3.9 | 3.2 | 3.7 | 3.0 |
| 140 | 4.3 | 3.8 | 3.1 | 3.5 | 2.9 |

*Table A.8: STD from the monoenergetic energies with the DECT scan, the algorithm used for reconstruction is iterative reconstruction. The values are from the Quasar phantom.*

| ***Energy [keV]*** | ***Inflated lung*** | ***Dense bone*** | ***Water*** | ***Inner bone*** | ***Polyethylene*** |
|---|---|---|---|---|---|
| 40 | 7.7 | 10.9 | 7.6 | 9.0 | 7.0 |
| 50 | 5.9 | 7.8 | 5.4 | 6.5 | 5.0 |
| 60 | 4.8 | 5.8 | 4.1 | 5.0 | 3.8 |
| 70 | 4.1 | 4.6 | 3.3 | 4.0 | 3.1 |
| 80 | 3.7 | 3.9 | 2.8 | 3.4 | 2.7 |
| 90 | 3.4 | 3.4 | 2.5 | 3.1 | 2.4 |
| 100 | 3.2 | 3.1 | 2.3 | 2.8 | 2.2 |
| 110 | 3.1 | 2.9 | 2.1 | 2.6 | 2.0 |
| 120 | 3.0 | 2.8 | 2.0 | 2.5 | 1.9 |
| 130 | 2.9 | 2.6 | 1.9 | 2.4 | 1.9 |
| 140 | 2.8 | 2.5 | 1.9 | 2.3 | 1.8 |

# A 3 Theoretical values

*Table A.9: Theoretical HU for the monochromatic images.*

| ***Energy [keV]*** | ***Blood70*** | ***Blood40*** | ***Brain*** | ***Water*** | ***Blood100*** | ***Adipose*** |
|---|---|---|---|---|---|---|
| 40 | 90 | 65 | 49 | -2 | 113 | -140 |
| 50 | 82 | 53 | 39 | -3 | 109 | -107 |
| 60 | 76 | 45 | 32 | -4 | 106 | -90 |
| 70 | 72 | 40 | 28 | -5 | 104 | -80 |
| 80 | 71 | 38 | 26 | -5 | 103 | -73 |
| 90 | 70 | 36 | 25 | -5 | 102 | -69 |
| 100 | 69 | 35 | 24 | -5 | 102 | -66 |
| 110 | 69 | 35 | 23 | -5 | 101 | -65 |
| 120 | 68 | 34 | 23 | -6 | 101 | -63 |
| 130 | 68 | 33 | 22 | -6 | 101 | -62 |
| 140 | 67 | 33 | 22 | -6 | 101 | -62 |

# A 4 SECT

*Table A.4: SECT HU, Gammex phantom 120 kVp.*

| ***Algorithm*** | ***Blood70*** | ***Blood40*** | ***Brain*** | ***Water*** | ***Blood100*** | ***Adipose*** |
|---|---|---|---|---|---|---|
| Back proj. | 79 | 46 | 34 | 2 | 106 | -67 |
| Iterative | 79 | 46 | 34 | 3 | 106 | -68 |

*Table A.5: Noise, Gammex phantom 120 kVp.*

| ***Algorithm*** | ***Blood70*** | ***Blood40*** | ***Brain*** | ***Water*** | ***Blood100*** | ***Adipose*** |
|---|---|---|---|---|---|---|
| Back proj. | 20.8 | 21.4 | 27.9 | 24.1 | 25.6 | 28.0 |
| Iterative | 12.4 | 12.8 | 16.3 | 14.4 | 15.0 | 16.5 |

*Table A.6: Mean HU for 120 kVp image of the Quasar phantom.*

| ***Algorithm*** | ***Inflated lung*** | ***Dense bone*** | ***Water*** | ***Inner bone*** | ***Polyethylene*** |
|---|---|---|---|---|---|
| Back proj. | -740 | 695 | 40 | 211 | -72 |
| Iterative | -740 | 695 | 40 | 211 | -72 |

*Table A.7: Noise for 120 kVp image of the Quasar phantom.*

| ***Algorithm*** | ***Inflated lung*** | ***Dense bone*** | ***Water*** | ***Inner bone*** | ***Polyethylene*** |
|---|---|---|---|---|---|
| Back proj. | 3.5 | 6.6 | 4.8 | 6.0 | 4.2 |
| Iterative | 2.5 | 5.3 | 3.1 | 3.9 | 2.5 |

# *Appendix B*

SIMULATION

*Table B.1: Simulated HU for 120 kVp for three different thickness.*

| ***Thickness [cm]*** | ***Soft tissue*** | ***Inflated lung*** | ***Cortical bone*** | ***Brain*** | ***Blood*** | ***Polyethylene*** | ***Adipose*** |
|---|---|---|---|---|---|---|---|
| 30 | 50 | -731 | 606 | 36 | 53 | -94 | -76 |
| 20 | 48 | -700 | 802 | 35 | 51 | -109 | -82 |
| 3 | 55 | -744 | 1123 | 41 | 62 | -150 | -111 |

*Table B.2: HU values for monoenergetic energies for different material from the CT calibration, with a thickness of 30 cm.*

| ***Energy [keV]*** | ***Soft tissue*** | ***Inflated lung*** | ***Cortical bone*** | ***Brain*** | ***Blood*** | ***Polyethylene*** | ***Adipose*** |
|---|---|---|---|---|---|---|---|
| 40 | 62 | -749 | 2522 | 47 | 73 | -211 | -152 |
| 50 | 58 | -750 | 1655 | 43 | 64 | -146 | -111 |
| 60 | 54 | -751 | 1171 | 39 | 59 | -110 | -89 |
| 70 | 53 | -751 | 960 | 38 | 57 | -95 | -80 |
| 80 | 52 | -752 | 723 | 37 | 54 | -77 | -69 |
| 90 | 52 | -752 | 636 | 36 | 53 | -71 | -65 |
| 100 | 51 | -752 | 543 | 36 | 53 | -63 | -61 |
| 110 | 51 | -752 | 517 | 36 | 52 | -61 | -59 |
| 120 | 51 | -752 | 489 | 36 | 52 | -59 | -58 |
| 130 | 51 | -752 | 460 | 36 | 52 | -57 | -56 |
| 140 | 51 | -752 | 429 | 35 | 51 | -55 | -55 |

# *Appendix C*

CT calibration

*Table C.3: CT calibration values for* Brilliance Big B *commissioned machine.*

| ***Brilliance Big B*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ***HU*** | -1000 | -992 | -976 | -480 | -96 | 48 | 128 | 528 |
| ***Mass density*** | 0.00121 | 0.00121 | 0.00121 | 0.5 | 0.95 | 1.05 | 1.1 | 1.35 |
| ***HU*** | 976 | 1488 | 1824 | 2224 | 2640 | 2832 | 2833 | 3096 |
| ***Mass density*** | 1.6 | 1.85 | 2.1 | 2.4 | 2.7 | 2.83 | 7.87 | 7.87 |

*Table C.4: CT calibration values for* Sandra *commissioned machine.*

| ***Sandra*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ***HU*** | -1000 | -992 | -976 | -480 | -68 | -52.4 | 1.06 | 37.3 |
| ***Mass density*** | 0.00121 | 0.00121 | 0.00121 | 0.5 | 0.94 | 0.965 | 1.021 | 1.03 |
| ***HU*** | 43.16 | 74.83 | 153.08 | 528 | 643.46 | 976 | 1488 | 1824 |
| ***Mass density*** | 1.06 | 1.095 | 1.1245 | 1.35 | 1.42 | 1.6 | 1.85 | 2.1 |
| ***HU*** | 2224 | 2640 | 2832 | 2833 | 3096 | | | |
| ***Mass density*** | 2.4 | 2.7 | 2.83 | 7.87 | 7.87 | | | |

# *Appendix D*

## D 1

*Table D.1: Delineation of target volumes. Also, dose restriction for the volumes used in RayStation is mentioned here*

| ***Volume*** | ***Description*** | ***Type*** | ***Colour*** | ***Dose calculation*** |
|---|---|---|---|---|
| Brain | | OAR | Brown | $V30\ Gy \leq 50\ \%$<br>$D_{max} \leq 68\ Gy$ |
| Brainstem | Delineate | OAR | Red brown | |
| BrainstemCore | Brainstem – 2 mm margin, except caudal | OAR | | |
| Brainstem_PRV | Brainstem + 3mm | OAR | | |
| BrainstemSurface | Brainstem – BrainstemCore | OAR | | |
| Eye_L/R | | OAR | Mangenta | $D_{max} \leq 30\ Gy$ |
| Eye_L/R_PVR | Eye_L/R + 3mm | OAR | | $D_{max} \leq 35\ Gy$ |
| Hippocampus_L/R | | OAR | Green | |
| Lacrimalgland_L/R | | OAR | Brown | |
| Lacrimalgland_L/R_PRV | Lacrimagland_L/R + 3 mm | OAR | | |
| Lense_L/R | | OAR | Blue | |
| Mandible | | OAR | Light blue | |
| OpticChiasm | | OAR | Light blue | $D_{max} \leq 54\ Gy$ |
| OpticChiasm_PRV | OpticChiasm + 3mm | OAR | | $D_{max} \leq 60\ Gy$ |
| OpticNerve_L/R | | OAR | Dark green | |
| OpticNerve_L/R_PRV | OpticNerve_L/R + 3mm | OAR | | $D_{max} \leq 60\ Gy$ |
| Parotide_L/R | | OAR | Orange | $D_{mean} \leq 26\ Gy$ |
| SpinalCord | Delineate | OAR | Light orange | |
| SpinalCord_PRV | SpinalCord + 3mm | OAR | | |
| SubmandGland_L/R | | OAR | Blue | $D_{mean} \leq 35\ Gy$ |

# *Appendix E*

Differenace between HU values.

## E 1 measured and simulate monochromatic HU values from DECT

*Table E.1: Correlation r*

| ***Energy [keV]*** | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 110 | 120 | 130 | 140 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Correlation [r]*** | 0.987 | 0.989 | 0.992 | 0.990 | 0.997 | 0.997 | 0.998 | 0.997 | 0.997 | 0.997 | 0.998 |

# *Appendix F*

HU and noise values in the Alderson phantoms

## F 1 HU values

*Table F.1:HU values for soft tissue in the whole Alderson. Backprojection.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| 40 | 60 | 53 | -4 | -44 | 2725 | 3058 | 3636 | 3886 | -1012 | -1017 | -1055 | -1028 |
| 50 | 39 | 33 | -3 | -30 | 1939 | 2180 | 2604 | 2764 | -1002 | -1004 | -1028 | -1005 |
| 60 | 28 | 22 | -3 | -22 | 1456 | 1640 | 1969 | 2074 | -996 | -996 | -1011 | -992 |
| 70 | 20 | 14 | -2 | -17 | 1156 | 1305 | 1576 | 1645 | -993 | -991 | -1001 | -983 |
| 80 | 15 | 10 | -3 | -14 | 963 | 1090 | 1323 | 1370 | -990 | -988 | -994 | -977 |
| 90 | 12 | 7 | -3 | -12 | 836 | 949 | 1156 | 1189 | -990 | -986 | -990 | -974 |
| 100 | 10 | 5 | -3 | -11 | 748 | 850 | 1041 | 1063 | -988 | -984 | -987 | -971 |
| 110 | 8 | 3 | -3 | -9 | 685 | 780 | 958 | 974 | -987 | -983 | -983 | -969 |
| 120 | 7 | 2 | -3 | -9 | 641 | 731 | 901 | 911 | -987 | -982 | -981 | -968 |
| 130 | 6 | 1 | -3 | -8 | 608 | 694 | 857 | 863 | -986 | -981 | -980 | -967 |
| 140 | 5 | 0 | -2 | -8 | 581 | 664 | 822 | 825 | -986 | -981 | -980 | -967 |

*Table F.2: HU values for soft tissue in the whole Alderson. Iterative reconstruction.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| 40 | 60 | 54 | -4 | -37 | 2725 | 3058 | 3636 | 3886 | -1012 | -1017 | -1051 | -1029 |
| 50 | 40 | 34 | -3 | -26 | 1939 | 2180 | 2604 | 2764 | -1002 | -1004 | -1026 | -1006 |
| 60 | 28 | 22 | -3 | -20 | 1456 | 1640 | 1969 | 2073 | -996 | -996 | -1010 | -992 |
| 70 | 20 | 15 | -2 | -15 | 1156 | 1305 | 1576 | 1645 | -994 | -991 | -1000 | -984 |
| 80 | 15 | 10 | -3 | -12 | 963 | 1090 | 1323 | 1370 | -991 | -988 | -992 | -977 |
| 90 | 12 | 7 | -3 | -10 | 836 | 949 | 1156 | 1189 | -989 | -985 | -988 | -974 |
| 100 | 9 | 5 | -2 | -9 | 748 | 850 | 1041 | 1063 | -988 | -984 | -985 | -972 |
| 110 | 8 | 3 | -2 | -8 | 686 | 780 | 958 | 974 | -987 | -982 | -983 | -970 |
| 120 | 7 | 2 | -2 | -7 | 642 | 731 | 901 | 911 | -986 | -982 | -981 | -969 |
| 130 | 6 | 1 | -2 | -7 | 608 | 694 | 857 | 863 | -986 | -981 | -980 | -968 |
| 140 | 5 | 1 | -2 | -6 | 581 | 664 | 822 | 825 | -986 | -980 | -980 | -967 |

*Table F.3: HU values for soft tissue in the slab Alderson. Backprojection.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| 40 | 46 | 47 | 17 | 26 | 3043 | 3505 | 3764 | 3812 | -982 | -978 | -1084 | -1045 |
| 50 | 26 | 27 | 7 | 14 | 2150 | 2486 | 2668 | 2718 | -988 | -986 | -1035 | -1014 |
| 60 | 12 | 16 | 1 | 6 | 1601 | 1859 | 1994 | 2046 | -992 | -991 | -1004 | -995 |
| 70 | 4 | 8 | -3 | 1 | 1260 | 1470 | 1576 | 1628 | -994 | -995 | -985 | -983 |
| 80 | -1 | 3 | -5 | -2 | 1041 | 1220 | 1308 | 1360 | -996 | -997 | -973 | -975 |
| 90 | -4 | 0 | -7 | -4 | 897 | 1056 | 1131 | 1184 | -997 | -998 | -965 | -970 |
| 100 | -6 | -3 | -8 | -5 | 797 | 941 | 1008 | 1061 | -998 | -999 | -960 | -966 |
| 110 | -8 | -5 | -9 | -6 | 726 | 860 | 921 | 974 | -998 | -1000 | -956 | -964 |
| 120 | -9 | -5 | -10 | -7 | 677 | 803 | 860 | 913 | -999 | -1000 | -953 | -962 |
| 130 | -10 | -6 | -10 | -7 | 638 | 760 | 813 | 866 | -999 | -1001 | -951 | -961 |
| 140 | -11 | -7 | -11 | -7 | 608 | 725 | 776 | 829 | -999 | -1001 | -949 | -960 |

*Table F.4: HU values for soft tissue in the slab Alderson. Iterative reconstruction.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| 40 | 47 | 46 | 19 | 26 | 3043 | 3505 | 3764 | 3812 | -981 | -976 | -1084 | -1047 |
| 50 | 26 | 27 | 8 | 14 | 2150 | 2486 | 2668 | 2718 | -987 | -985 | -1034 | -1015 |
| 60 | 13 | 15 | 2 | 6 | 1601 | 1859 | 1995 | 2046 | -992 | -991 | -1004 | -995 |
| 70 | 5 | 7 | -2 | 1 | 1260 | 1470 | 1577 | 1628 | -994 | -994 | -985 | -983 |
| 80 | 0 | 2 | -5 | -2 | 1041 | 1220 | 1308 | 1360 | -996 | -996 | -973 | -975 |
| 90 | -4 | -1 | -7 | -4 | 898 | 1056 | 1131 | 1184 | -996 | -998 | -965 | -970 |
| 100 | -6 | -3 | -8 | -5 | 797 | 941 | 1008 | 1061 | -997 | -999 | -960 | -967 |
| 110 | -8 | -4 | -9 | -6 | 726 | 860 | 921 | 974 | -998 | -999 | -956 | -964 |
| 120 | -9 | -5 | -9 | -7 | 676 | 803 | 859 | 913 | -998 | -1000 | -953 | -962 |
| 130 | -10 | -6 | -10 | -7 | 638 | 760 | 812 | 866 | -999 | -1000 | -951 | -961 |
| 140 | -11 | -7 | -10 | -7 | 608 | 725 | 775 | 829 | -999 | -1001 | -949 | -960 |

# F 2 Noise

*Table F.5: Noise values for soft tissue in the whole Alderson. Backprojection.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| 40 | 29.5 | 22.9 | 18.9 | 33.2 | 1028.5 | 954.7 | 510.6 | 119.5 | 22.6 | 21.3 | 42.3 | 45.3 |
| 50 | 21.6 | 16.7 | 12.9 | 23.2 | 732.8 | 684.2 | 361.8 | 93.1 | 16.0 | 15.2 | 27.7 | 29.0 |
| 60 | 17.0 | 13.6 | 9.6 | 17.9 | 550.8 | 517.9 | 270.6 | 76.9 | 12.1 | 11.6 | 18.7 | 19.3 |
| 70 | 14.0 | 12.2 | 7.9 | 14.6 | 438.0 | 414.9 | 214.0 | 67.1 | 9.6 | 9.3 | 13.3 | 13.3 |
| 80 | 11.6 | 10.4 | 7.7 | 12.7 | 365.7 | 348.6 | 178.0 | 60.7 | 8.4 | 7.4 | 10.0 | 9.9 |
| 90 | 10.2 | 9.2 | 6.8 | 11.6 | 318.0 | 305,1 | 154.2 | 56.6 | 7.2 | 6.8 | 7.8 | 7.5 |
| 100 | 9.3 | 8.4 | 6.4 | 10.9 | 285.0 | 274.7 | 137.9 | 53.6 | 6.8 | 6.1 | 6.5 | 6.2 |
| 110 | 8.5 | 7.8 | 6.1 | 10.0 | 261.6 | 253.2 | 126.3 | 51.6 | 6.5 | 5.6 | 4.8 | 5.6 |
| 120 | 8.0 | 7.2 | 5.7 | 9.4 | 245.1 | 238.3 | 118.2 | 50.2 | 6.1 | 5.4 | 4.1 | 5.1 |
| 130 | 7.7 | 7.1 | 5.4 | 8.9 | 232.5 | 226.9 | 1119 | 49.1 | 6.0 | 5.1 | 3.8 | 4.8 |
| 140 | 7.3 | 6.8 | 5.3 | 8.5 | 222.4 | 217.8 | 107.,0 | 48.2 | 5.9 | 4.9 | 3.5 | 4.8 |

*Table F.6: Noise values for soft tissue in the whole Alderson. Iterative reconstruction.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| 40 | 21.3 | 14.7 | 13.9 | 22.5 | 1028.5 | 954.8 | 510.7 | 119.0 | 14.8 | 15.0 | 36.9 | 39.9 |
| 50 | 15.6 | 10.9 | 9.9 | 16.5 | 732.89 | 684.3 | 362.0 | 92.9 | 11.0 | 11.0 | 23.9 | 25.4 |
| 60 | 11.7 | 9.0 | 7.5 | 13.4 | 551.0 | 517.9 | 270.6 | 76.8 | 8.6 | 8.2 | 16.1 | 16.9 |
| 70 | 9.5 | 7.5 | 6.2 | 11.3 | 438.1 | 414.9 | 214.2 | 67.0 | 7.3 | 6.2 | 11.4 | 11.6 |
| 80 | 8.1 | 6.6 | 5.8 | 9.7 | 365.7 | 348.8 | 177.9 | 60.6 | 6.6 | 5.6 | 7.8 | 8.6 |
| 90 | 6.9 | 5.8 | 5.2 | 8.5 | 318.2 | 305.4 | 154.3 | 56.4 | 6.1 | 4.9 | 6.0 | 6.4 |
| 100 | 6.3 | 5.3 | 4.7 | 7.6 | 285.0 | 275.1 | 137.8 | 53.6 | 5.9 | 4.4 | 4.8 | 5.4 |
| 110 | 5.8 | 4.9 | 4.3 | 6.9 | 261.5 | 253.7 | 126.3 | 51.5 | 5.6 | 4.2 | 3.9 | 4.8 |
| 120 | 5.4 | 4.7 | 4.0 | 6.6 | 244.8 | 238.5 | 118.1 | 50.2 | 5.4 | 3.9 | 3.4 | 4.5 |
| 130 | 5.2 | 4.5 | 3.9 | 6.3 | 232.2 | 226.8 | 111.9 | 49.1 | 5.3 | 3.7 | 3.0 | 4.5 |
| 140 | 5.0 | 4.3 | 3.8 | 5.9 | 222.3 | 217.9 | 107.0 | 48.2 | 5.1 | 3.6 | 2.9 | 4.4 |

*Table F.7: Noise values for soft tissue in the slab Alderson. Backprojection.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| 40 | 23.8 | 21.5 | 25.5 | 15.3 | 690.1 | 525.7 | 86.6 | 245.4 | 17.8 | 23.7 | 36.5 | 23.8 |
| 50 | 17.1 | 15.4 | 17.7 | 11.4 | 491.8 | 374.2 | 62.2 | 176.7 | 13.0 | 17.3 | 25.0 | 15.7 |
| 60 | 13.6 | 12.5 | 13.6 | 9.2 | 370.7 | 281.8 | 47.3 | 135.0 | 10.1 | 13.4 | 18.2 | 11.2 |
| 70 | 11.8 | 10.1 | 11.2 | 7.9 | 296.1 | 224.9 | 38.2 | 109.6 | 8.3 | 11.0 | 14.1 | 8.8 |
| 80 | 10.0 | 8.6 | 9.3 | 7.1 | 248.6 | 188.7 | 32.2 | 93.9 | 7.3 | 9.7 | 11.5 | 7.9 |
| 90 | 9.0 | 7.9 | 8.1 | 6.8 | 217.8 | 165.3 | 28.2 | 83.6 | 6.4 | 8.5 | 10.0 | 7.3 |
| 100 | 8.0 | 7.1 | 7.4 | 6.3 | 196.6 | 149.2 | 25.7 | 76.9 | 6.0 | 7.7 | 8.9 | 7.1 |
| 110 | 7.5 | 6.5 | 6.7 | 6.0 | 182.0 | 137.7 | 23.7 | 72.0 | 5.7 | 7.3 | 8.3 | 7.2 |
| 120 | 7.1 | 6.2 | 6.4 | 5.8 | 171.6 | 129.9 | 22.4 | 68.8 | 5.5 | 7.1 | 8.0 | 7.1 |
| 130 | 6.76 | 5.9 | 6.1 | 5.5 | 163.7 | 123.9 | 21.3 | 66.3 | 5.3 | 6.8 | 7.6 | 7.0 |
| 140 | 6.7 | 5.8 | 5.9 | 5.3 | 157.6 | 119.3 | 20.6 | 64.2 | 5.1 | 6.4 | 7.4 | 7.1 |

*Table F.8: Noise values for soft tissue in the slab Alderson. Iterative reconstruction.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| 40 | 16.6 | 15.7 | 17.6 | 9.8 | 689.9 | 525.5 | 86.2 | 245.3 | 11.3 | 17.2 | 30.2 | 17.8 |
| 50 | 12.4 | 11.3 | 13.2 | 7.8 | 491.8 | 374.1 | 61.8 | 176.6 | 8.4 | 12.5 | 20.9 | 11.9 |
| 60 | 10.0 | 8.7 | 10.0 | 6.7 | 370.8 | 281.7 | 47.1 | 134.8 | 6.7 | 10.0 | 15.3 | 8.7 |
| 70 | 8.3 | 7.0 | 7.9 | 6.0 | 296.0 | 224.8 | 37.9 | 109.6 | 5.7 | 8.3 | 12.1 | 7.4 |
| 80 | 7.1 | 6.1 | 6.8 | 5.1 | 248.5 | 188.7 | 32.0 | 93.9 | 5.1 | 7.4 | 10.2 | 6.9 |
| 90 | 6.3 | 5.3 | 6.0 | 4.9 | 218.1 | 165.3 | 28.2 | 83.6 | 4.7 | 6.8 | 8.9 | 6.8 |
| 100 | 5.8 | 4.7 | 5.4 | 4.5 | 196.9 | 149.2 | 25.5 | 76.8 | 4.5 | 6.2 | 8.2 | 6.8 |
| 110 | 5.5 | 4.4 | 5.0 | 4.3 | 182.1 | 137.7 | 23.6 | 72.0 | 4.4 | 6.1 | 7.7 | 6.7 |
| 120 | 5.1 | 4.3 | 4.6 | 4.1 | 171.5 | 129.9 | 22.3 | 68.8 | 4.3 | 5.7 | 7.5 | 6.8 |
| 130 | 4.9 | 3.9 | 4.4 | 4.0 | 163.7 | 123.9 | 21.3 | 66.3 | 4.3 | 5.5 | 7.2 | 6.9 |
| 140 | 4.7 | 3.9 | 4.2 | 3.7 | 157.6 | 119.3 | 20.5 | 64.3 | 4.1 | 5.3 | 7.0 | 7.0 |

# F 3 SECT HU AND NOISE

*Table F.9: HU values for 120 kVp image of the whole phantom, backprojection and iterative.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| ***BP*** | 19 | 23 | 24 | 28 | 1340 | 1399 | 1442 | 1521 | -978 | -994 | -983 | -966 |
| ***IT*** | 18 | 22 | 24 | 27 | 1340 | 1399 | 1442 | 1521 | -978 | -994 | -983 | -967 |

*Table F.10: HU values for 120 kVp image of the slab phantom, backprojection and iterative reconstruction*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| ***BP*** | 12 | 19 | 2 | 5 | 1263 | 1387 | 1480 | 1549 | -982 | -977 | -972 | -978 |
| ***IT*** | 12 | 18 | 3 | 4 | 1263 | 1387 | 1480 | 1549 | -982 | -977 | -972 | -978 |

*Table F.11: Noise values for 120 kVp image of the whole phantom, backprojection and iterative reconstruction.*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | ST – 1 | ST – 2 | ST – 3 | ST – 4 | B – 1 | B – 2 | B – 3 | B – 4 | A – 1 | A – 2 | A – 3 | A – 4 |
| ***BP*** | 8.7 | 8.0 | 8.1 | 12.9 | 190.6 | 112.1 | 143.2 | 25.1 | 5.5 | 8.4 | 6.1 | 8.3 |
| ***IT*** | 5.6 | 5.9 | 6.9 | 9.4 | 190.8 | 112.1 | 143.2 | 25.0 | 4.2 | 7.7 | 4.9 | 7.5 |

*Table F.12: Noise values for 120 kVp image of the slab phantom, backprojection and iterative reconstruction*

| | *Soft tissue* | | | | *Bone* | | | | *Air* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Energy [keV]** | *ST – 1* | *ST – 2* | *ST – 3* | *ST – 4* | *B – 1* | *B – 2* | *B – 3* | *B – 4* | *A – 1* | *A – 2* | *A – 3* | *A – 4* |
| ***BP*** | 8.2 | 9.0 | 10.7 | 8.7 | 201.1 | 178.6 | 47.8 | 44.3 | 5.9 | 4.8 | 7.5 | 11.9 |
| ***IT*** | 6.0 | 6.2 | 7.3 | 6.0 | 201.1 | 178.7 | 47.6 | 44.2 | 5.0 | 4.0 | 7.2 | 10.2 |

# *Appendix G*

Proton treatment plan

*Table G.1: Brilliance Bib B CT*

| | *Nasopharynxx tumour* | | | *Pharynx tumour* | | | |
|---|---|---|---|---|---|---|---|
| | CTV | OpticChiasm | OpricNerve_L | CTV | Parotide_R | SpinalCord | SubmandGland |
| ***Energy [kVp]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** |
| 120 | 65.71 | 0.14 | 0.14 | 65.11 | 0.38 | 0.01 | 3.97 |
| ***Energy [keV]*** | | | | | | | |
| 40 | 18.29 | 0.13 | 0.02 | 35.17 | 0.35 | 0.01 | 4.58 |
| 60 | 65.48 | 0.16 | 0.11 | 64.64 | 0.39 | 0.01 | 4.14 |
| 80 | 65.78 | 0.16 | 0.15 | 65.28 | 0.37 | 0.01 | 3.98 |
| 100 | 65.63 | 0.17 | 0.15 | 65.51 | 0.38 | 0.01 | 4.1 |
| 120 | 65.90 | 0.15 | 0.16 | 65.41 | 0.38 | 0.01 | 4.03 |
| 140 | 65.87 | 0.17 | 0.17 | 65.66 | 0.37 | 0.02 | 4.01 |

*Table G.2: Sandra CT*

| | *Nasopharynx tumour* | | | *Pharynx tumour* | | | |
|---|---|---|---|---|---|---|---|
| | CTV | OpticChiasm | OpricNerve_L | CTV | Parotide_R | SpinalCord | SubmandGland |
| ***Energy [kVp]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** | ***D99 [Gy]*** |
| 120 | 65.71 | 0.13 | 0.11 | 65.14 | 0.37 | 0.01 | 4.00 |
| ***Energy [keV]*** | | | | | | | |
| 40 | 14.42 | 0.18 | 0.01 | 35.36 | 0.35 | 0.02 | 4.06 |
| 60 | 65.37 | 0.13 | 0.13 | 64.57 | 0.39 | 0.01 | 4.06 |
| 80 | 65.51 | 0.14 | 0.12 | 65.17 | 0.39 | 0.01 | 4.02 |
| 100 | 65.71 | 0.16 | 0.14 | 65.32 | 0.39 | 0.02 | 4.07 |
| 120 | 65.79 | 0.18 | 0.17 | 65.45 | 0.38 | 0.02 | 4.06 |
| 140 | 65.79 | 0.16 | 0.17 | 65.57 | 0.37 | 0.02 | 4.08 |

# *Appendix H*

Python code to simulate x-ray spectrum and find HU. Calculation of SPR. All plots are set to False as default.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import zoomed_inset_axes
from mpl_toolkits.axes_grid1.inset_locator import mark_inset
import matplotlib.lines as mlines
from scipy import interpolate
import xlrd

'''A program to calculate the Hounsfield Unit that we get from an acquisition vs
the stopping power ratio for a proton. '''

#Variables of the energy
kv      = 120                                   # maximum energy [keV]
hv      = 10 + np.arange(kv - 9)                # an array with energies up to max [keV]
hv_max = kv                                     # set the max energy [keV]
K       = 100                                   # a konstant in Kramers
specter
x       = 30                                    # thickness of the object
[cm]

# Constants for the filter
x_filter        = 0.39                          # thickness of the filter
mu_pre_filt = 1                                 # filtering the first photon
specter

# Kramers specter, info about how the x-ray specter from brehmsstrahlung
psi_spec_CT = K * (hv_max - hv)
#psi_tot = np.sum(psi)                          #area under the curve

# Normally used: 2.5 mm Al or 0.1-0.9 mm Cu

#Files containing energy and attenuation values
Al_file         = 'aluminum-attenuation.txt'
Cu_file         = 'copper-attenuation.txt'
Water_file      = 'water-attenuation.txt'
Lung_file       = 'lung-attenuation.txt'
Soft_t_file = 'softtissue-attenuation.txt'
Cort_file       = 'cortical-attenuation.txt'
Brain_file      = 'brain-attenuation.txt'
Blood_file      = 'blood-attenuation.txt'
Poly_file       = 'polyethylene-attenuation.txt'
Adip_file       = 'Adipose-attenuation.txt'

# Densities with unit [g/cm³]
Al_dens         = 2.7                           # Alumnium
Cu_dens         = 8.96                          # Copper
```

```
Water_dens    = 1.00                                    # Water
Lung_dens     = 0.25                                    # Lung
Soft_t_dens = 1.060                                  # Soft tissue
Cort_dens     = 1.420                                   # Cortical bone          , have used
1.609, NIST 1.920
Brain_dens    = 1.040                                   # Brain
Blood_dens    = 1.060                                   # Blood
Poly_dens     = 9.30e-1                                 # Polyethylene
Adip_dens     = 9.5e-1                                  # Adipose

# A function to implement datasets from text files
def read_file(textfile1, density):
        hv_file         = []                                    # [MeV]
        murho_file      = []                                    # mu/rho [cm^2/g]


        #Read in the file soft_tissue.txt, first two lines
        with open(textfile1) as myfile: #'soft_tissue.txt'
                infile = myfile.readlines()             # read the values

                for line in infile:

                        values = line.split()                   # splits the values with ' '
                        value_hv = float(values[0])         # the numbers at index 0 float
                        hv_file.append(value_hv)                # appends the first collum

                        value_murho = float(values[1])          # index 1 to float
                        murho_file.append(value_murho)          # appends the 2. collum

        myfile.close

        # Change from list to an array
        hv_file = np.array(hv_file) * 1000.                             # [keV]
        mu_file = np.array(murho_file) * density                        # [cm-1]
        #print(density)
        return hv_file, mu_file

# A function to implement data from text file
def element(textfile, density):
        # Adipode tissue ICRU- 44 density: 9.500E-01
        element_hv              = []
        # [MeV]
        element_murho           = []
        # mu/rho [cm^2/g]

        #Read in the file soft_tissue.txt, first two lines
        with open(textfile) as myfile:
                infile = myfile.readlines()                                     # read the
values
```

```
            for line in infile:

                values = line.split()                    # splits the values with ' '
                value_hv = float(values[0])          # the numbers at index 0 float
                element_hv.append(value_hv)          # appends the first collum

                value_murho = float(values[1])       # index 1 to float
                element_murho.append(value_murho)    # appends the 2. collum

        myfile.close

        # Change from list to an array
        element_hv = np.array(element_hv) * 1000.                    # [keV]
        element_mu = np.array(element_murho) *  density          # [cm^-1]
        #print(density)

        return element_hv, element_mu

def interpolation(tissue, energy):
        x1      = tissue[0]
        # energy
        y1      = tissue[1]
        # attenuation

        f       = interpolate.interp1d(x1, y1, kind='linear')          # interpolation

        xnew = energy                                                  # the energy spectra
we want to use

        # The interpolation gives us the attenuation values for the array
        # with energy. It uses the attenuation and enegry from given data
        interp_mu = f(xnew)                                  # what we want to interpolate

        return interp_mu
        # Kramer´s law tells us how the x-ray spectra is produced by an electron
        # hitting a solid target. We want to use the whole spectra,
        # not a monoenergetic value.

def pre_filtration(thickness, filt, energy, typ):
        mu              = interpolation(filt, energy)
        x_filt  = thickness

        if typ == 0:
                # make a spectrum of the energy (original CT)
                psi                     = K * ( np.max(energy) - energy)
                psi_prefilt     = psi * np.exp( - mu * x_filt )
        else:
                psi             = energy
                psi_prefilt = psi * np.exp( - mu * x_filt )
```

```
        psi_prefilt_tot = np.sum(psi_prefilt)
        psi_tot                 = np.sum(psi)


        # Returns both filtered and non-filtered psi
        return psi, psi_prefilt

def filtration(interp_mu, filt, thickness, energy, typ):
        #psi_interp = psi_prefilt
        psi_interp = pre_filtration(thickness, filt, energy, typ)

        # we know that the specter will loose enegry with e^-mu*x
        psi_interp_filter = psi_interp[1] * np.exp( - interp_mu * x)
        psi_interp_nofilter = psi_interp[0] * np.exp( - interp_mu * x)

        # To be able to calculate the mean attenuation we have to find all the
        # photons before hitting the patient divided by all the photons hitting the
        # detector.
        psi_interp_tot              = np.sum(psi_interp[1])
        psi_interp_filter_tot   = np.sum(psi_interp_filter)

        mu_mean_interp = (np.log( psi_interp_tot / psi_interp_filter_tot )) / x

        return psi_interp[1], psi_interp_filter, \
        psi_interp_tot, psi_interp_filter_tot\
        , mu_mean_interp, psi_interp_nofilter

def water_filtration(interp_mu, energy, typ):

        #psi_interp = pre_filtration(thickness, filt)
        # we know that the specter will loos enegry with e^-mu*x
        if typ == 0:
                # make a spectrum of the energy (original CT image)
                psi                                     = K * (hv_max - energy)
                psi_interp_filter               = psi * np.exp( - interp_mu * x)
        else:
                psi = energy
                psi_interp_filter               = psi * np.exp( - interp_mu * x)

        # To be able to calculate the mean attenuation we have to find all the
        # photons before hitting the patient divided by all the photons hitting the
        # detector.
        #psi_interp_tot = np.sum(psi_interp[0])
        #psi_interp_filter_tot = np.sum(psi_interp_filter)
        psi_interp_tot              = np.sum(psi)
        psi_interp_filter_tot   = np.sum(psi_interp_filter)

        mu_mean_interp = (np.log( psi_interp_tot / psi_interp_filter_tot )) / x

        return psi_interp_filter, psi_interp_tot, psi_interp_filter_tot\
```

```
        , mu_mean_interp

#########################################################################
#####
#########################################################################
#####
#########################################################################
#####
#####
                                        #####
#####           FIND THE ATTENUATION VALUES FOR DIFFERENT MEDIUMS AND
PLOT    #####
#####
                                        #####
#########################################################################
#####
#########################################################################
#####
#########################################################################
#####

#########################################################################
#####
#####                                                W A T E R
                                    #####
#########################################################################
#####
water_calculation = True
if water_calculation:

        water_mu_inter = interpolation(element(Water_file, Water_dens), hv )
        mu_w = filtration(water_mu_inter, element(Al_file, Al_dens), x_filter,\
                          hv, 0)

        mu_water_list = []

        ## A for loop to run through the keV between 20-140, steps in 10 keV
        for i in range(10,140,10):

                en = i + 10
                water_mu_inter_2 = interpolation(element(Water_file, Water_dens), en )
                mu_w_2 = filtration(water_mu_inter_2, element(Al_file, Al_dens), \
                        x_filter, en, 1)


                mu_water_list.append(mu_w_2[4])

        mu_water_list = np.array(mu_water_list)

        print_w = False
```

```
        if print_w:
                print(mu_water_list)

##########################################################################
#####
#####                                S O F T  T I S S U E
                  #####
##########################################################################
#####

soft_tissue_on = True
if soft_tissue_on:

        Aluminum_filter_soft = True
        if Aluminum_filter_soft:
                # Soft tissue with aluminum filter - Krisin: 3.99 mm ved 75kV
                values_soft = filtration( interpolation( read_file(Soft_t_file, \
                        Soft_t_dens), hv ), element(Al_file, Al_dens), x_filter, hv, 0)

                mu_soft_list  = []

                for i in range(10,140,10):
                        en = 10 + i
                        fil_soft  = read_file(Soft_t_file, Soft_t_dens)
                        inte_soft = interpolation(fil_soft, en)
                        mu_soft = filtration(inte_soft, element(Al_file, Al_dens),\
                                x_filter, en, 1)

                        mu_soft_list.append(mu_soft[4])

                mu_soft_list = np.array(mu_soft_list)

                plot_soft_t = False
                if plot_soft_t:
                        plt.step(hv, psi_spec_CT, label='Original')
                        plt.plot(hv, psi_spec_CT,'-', color='grey', alpha=0.3)
                        plt.step(hv, values_soft[0], label='Aluminium filter')
                        plt.plot(hv, values_soft[0],'-', color='grey', alpha=0.3 )
                        plt.step(hv, values_soft[1] * 150, label='Both filters * 150')
                        plt.plot(hv, values_soft[1] * 150, '-', color='grey', alpha=0.3)
                        plt.step(hv, values_soft[-1] * 150, label='Body only * 150')
                        plt.plot(hv, values_soft[-1] * 150, '-', color='grey', alpha=0.3)
                        plt.legend()
                        plt.title('Soft tissue')
                        plt.xlabel('h$ \\nu $ [keV]')
                        plt.ylabel('$ \psi $ (Photon fluence)')
                        plt.grid(True, color='gainsboro')
                        plt.show()

                mu_mean_after   = values_soft[4]
```

```
            mu_mean                 = mu_soft_list

            # Now that we are able to find the attenuation through the body and filter
            # we want to calculate the Hounsfield Unit
            HU                  = 1000 * ( ( mu_mean_after - mu_w[-2] ) / mu_w[-2] )
            HU_soft_2 = 1000 * ( ( mu_mean - mu_water_list ) / mu_water_list )

        print_s = False
        if print_s:
            print('The CT number is %.3f HU for Soft tissue.' % HU)
            print(HU_soft_2)
            print('The CT number is %.3f HU for Soft tissue 120 kV.' % HU)
            print(HU)

##########################################################################
#####
#####                                                 L U N G
                                        #####
##########################################################################
#####

Lung_on = True
if Lung_on:

        Lung_aluminum = True
        if Lung_aluminum:
            lung_t   = read_file(Lung_file, Lung_dens)
            int_value = interpolation(lung_t, hv)
            mu_lung = filtration(int_value, element(Al_file, Al_dens),\
             x_filter, hv, 0)


            mu_lung_list  = []

            for i in range(10,140,10):
                #en    = 10 + np.arange((i+10) - 9)
                en = 10 + i
                fil_lung  = read_file(Lung_file, Lung_dens)
                inte_lung = interpolation(fil_lung, en)
                mu_lung_2 = filtration(inte_lung, element(Al_file, Al_dens),\
                 x_filter, en, 1)

                mu_lung_list.append(mu_lung_2[4])

            mu_lung_list = np.array(mu_lung_list)

            plot_lung = False
            if plot_lung:
                plt.plot(hv, psi_spec_CT, label='Original')
                plt.plot(hv, mu_lung[0], label='Aluminium filter')
```

```
                    plt.plot(hv, mu_lung[1], label='Both filters')
                    plt.plot(hv, mu_lung[-1], label='Body')
                    plt.legend()
                    plt.title('Lung')
                    plt.xlabel('$h\\nu$ [keV]')
                    plt.ylabel('$ \psi $ (Number of photons)')
                    plt.show()

            mu_mean_al_lung = mu_lung[4]
            mu_mean_l    = mu_lung_list

            HU_al_lung  = 1000 * ( ( mu_mean_al_lung - mu_w[-2] ) / mu_w[-2] )
            HU_lung       = 1000 * ( ( mu_mean_l - mu_water_list ) / mu_water_list )

    print_l = False
    if print_l:
            print('The CT number is %.3f HU for inflated lungs.' % HU_al_lung)
            print(HU_lung)
            print('The CT number is %.3f HU for inflated lungs at 120 kV.' %
HU_al_lung)
            print(HU_al_lung)


#############################################################################
#####
#####                              C O R T I C A L  B O N E
              #####
#############################################################################
#####

Cortical_on = True
if Cortical_on:

    Cortical_aluminum = True
    if Cortical_aluminum:
            cortical_t      = read_file(Cort_file, Cort_dens)
            int_value       = interpolation(cortical_t, hv)
            mu_cortical = filtration(int_value, element(Al_file, Al_dens),\
             x_filter, hv, 0)

            plot_cortical = False
            if plot_cortical:
                    plt.plot(hv, psi, label='Original psi')
                    plt.plot(hv, mu_cortical[0], label='Filtered through Al')
                    plt.plot(hv, mu_cortical[1], label='Both filters')
                    plt.plot(hv, mu_cortical[-1], label='Only bodyfilter')
                    plt.legend()
                    plt.title('')
                    plt.xlabel('energy [hv]')
                    plt.ylabel('$ \psi $')
```

```
                plt.show()

        mu_cort_list = []

        for i in range(10,140,10):
                #en = 10 + np.arange((i+10) - 9)
                en = 10 + i
                fil_cort = read_file(Cort_file, Cort_dens)
                inte_cort = interpolation(fil_cort, en)
                mu_cort = filtration(inte_cort, element(Al_file, Al_dens),\
                 x_filter, en, 1)

                mu_cort_list.append(mu_cort[4])

        mu_cort_list            = np.array(mu_cort_list)

        mu_mean_cort                    = mu_cort_list
        mu_mean_cortical    = mu_cortical[4]

        HU_cort_list  = 1000 * ( ( mu_mean_cort - mu_water_list ) /\
                                        mu_water_list )
        HU_cort                 = 1000 * ( ( mu_mean_cortical - mu_w[-2] ) /\
                                        mu_w[-2] )

        #print(HU_cort_list)
        print_c = False
        if print_c:
                print('The CT number is %.3f HU for cortical bone.' % HU_cort)
                print(HU_cort_list)
                print('The CT number is %.3f HU for cortical bone at 120 kV.' %
HU_cort)
                print(HU_cort)

                # The CT do not use CT number above 1000 HU, here we will check
since
                # we know bone has a high attenuation
                if HU_cort > 1000:
                        HU_cort = 1000
                        print('HU Cortial is larger than 1000 HU')

        for i in range(len(HU_cort_list)):
                if HU_cort_list[i] > 3071:
                        HU_cort_list[i] = 3071

        #print(HU_cort_list)

##########################################################################
#####
#####                                             B R A I N
                        #####
```

```
#########################################################################
#####

Brain_on = True
if Brain_on:

        hv_mu_brain = read_file(Brain_file, Brain_dens)

        int_value       = interpolation(hv_mu_brain, hv)
        mu_brain_v      = filtration(int_value, element(Al_file, Al_dens),\
                                        x_filter, hv, 0)

        mu_brain        = []

        for i in range(10, 140, 10):
                #en = 10 + np.arange((i+10) - 9)
                en = 10 + i
                mu_interp_brain = interpolation(hv_mu_brain, en)
                brain = filtration(mu_interp_brain, element(Al_file, Al_dens),\
                                x_filter, en, 1)

                mu_brain.append(brain[4])

        mu_brain        = np.array(mu_brain)


        # Hounsfield unit

        HU_brain_range      = 1000 * ((mu_brain - mu_water_list) / mu_water_list)
        HU_brain            = 1000 * ((mu_brain_v[4] - mu_w[-2]) / mu_w[-2])

        print_br = False
        if print_br:
                print('The CT number is %.3f HU for brain.' % HU_brain)
                print(HU_brain_range)
                print('The CT number is %.3f HU for brain for 120 kV.' % HU_brain)
                print(HU_brain)

        plot_brain = False
        if plot_brain:
                plt.plot(hv, psi, label='Original psi')
                plt.plot(hv, mu_brain_v[0], label='Filtered through Al')
                plt.plot(hv, mu_brain_v[1], label='Both filters')
                plt.plot(hv, mu_brain_v[-1], label='Only bodyfilter')
                plt.legend()
                plt.title('')
                plt.xlabel('energy [hv]')
                plt.ylabel('$ \psi $')
                plt.show()
```

```
#############################################################################
#####
#####                                        B L O O D
                                  #####
#############################################################################
#####

Blood_on = True
if Blood_on:

        hv_mu_blood = read_file(Blood_file, Blood_dens)
        mu_blood    = []

        for i in range(10, 140, 10):
                en = 10 + i
                #en = 10 + np.arange((i+10) - 9)
                mu_interp_blood = interpolation(hv_mu_blood, en)
                blood = filtration(mu_interp_blood, element(Al_file, Al_dens),\
                                x_filter, en, 1)

                mu_blood.append(blood[4])

        mu_blood    = np.array(mu_blood)

        int_value       = interpolation(hv_mu_blood, hv)
        mu_blood_v  = filtration(int_value, element(Al_file, Al_dens),\
                                x_filter, hv, 0)

        # Hounsfield unit

        HU_blood_range      = 1000 * ((mu_blood - mu_water_list) / mu_water_list)
        HU_blood            = 1000 * ((mu_blood_v[4] - mu_w[-2]) / mu_w[-2])

        print_bl = False
        if print_bl:
                print('The CT number is %.3f HU for blood.' % HU_blood)
                print(HU_blood_range)
                print('The CT number is %.3f HU for blood for 120 kV.' % HU_blood)
                print(HU_blood)

        plot_blood = False
        if plot_blood:
                plt.plot(hv, psi, label='Original psi')
                plt.plot(hv, mu_blood_v[0], label='Filtered through Al')
                plt.plot(hv, mu_blood_v[1], label='Both filters')
                plt.plot(hv, mu_blood_v[-1], label='Only bodyfilter')
                plt.legend()
                plt.title('sn - Al - blood')
                plt.xlabel('energy [hv]')
                plt.ylabel('$ \psi $')
```

```
                plt.show()

##########################################################################
#####
#####                                P O L Y E T H Y L E N E
                            #####
##########################################################################
#####

Poly_on = True
if Poly_on:

        hv_mu_poly   = read_file(Poly_file, Poly_dens)
        mu_poly      = []

        for i in range(10, 140, 10):
                en = 10 + i
                #en = 10 + np.arange((i+10) - 9)
                mu_interp_poly = interpolation(hv_mu_poly, en)
                poly = filtration(mu_interp_poly, element(Al_file, Al_dens),\
                                x_filter, en, 1)

                mu_poly.append(poly[4])

        mu_poly      = np.array(mu_poly)

        int_value    = interpolation(hv_mu_poly, hv)
        mu_poly_v    = filtration(int_value, element(Al_file, Al_dens),\
                                x_filter, hv, 0)

        # Hounsfield unit

        HU_poly_range       = 1000 * ((mu_poly - mu_water_list) / mu_water_list)
        HU_poly             = 1000 * ((mu_poly_v[4] - mu_w[-2]) / mu_w[-2])

        print_p = False
        if print_p:
                print('The CT number is %.3f HU for polyethylene.' % HU_poly)
                print(HU_poly_range)
                print('The CT number is %.3f HU for polyethylene for 120 kV.' % HU_poly)
                print(HU_poly)

        plot_blood = False
        if plot_blood:
                plt.plot(hv, psi, label='Original psi')
                plt.plot(hv, mu_poly_v[0], label='Filtered through Al')
                plt.plot(hv, mu_poly_v[1], label='Both filters')
                plt.plot(hv, mu_poly_v[-1], label='Only bodyfilter')
                plt.legend()
                plt.title('see Al - polyethylene')
```

```
            plt.xlabel('energy [hv]')
            plt.ylabel('$ \psi $')
            plt.show()

#############################################################################
#####
#####                                         A D I P O S E
                        #####
#############################################################################
#####

Adip_on = True
if Adip_on:

        hv_mu_adip   = read_file(Adip_file, Adip_dens)
        mu_adip      = []

        for i in range(10, 140, 10):
                en = 10 + i
                #en = 10 + np.arange((i+10) - 9)
                mu_interp_adip = interpolation(hv_mu_adip, en)
                adip = filtration(mu_interp_adip, element(Al_file, Al_dens),\
                                x_filter, en, 1)

                mu_adip.append(adip[4])

        mu_adip      = np.array(mu_adip)

        int_value    = interpolation(hv_mu_adip, hv)
        mu_adip_v    = filtration(int_value, element(Al_file, Al_dens),\
                                x_filter, hv, 0)

        # Hounsfield unit
        HU_adip_range = 1000 * ((mu_adip - mu_water_list) / mu_water_list)
        HU_adip = 1000 * ((mu_adip_v[4] - mu_w[-2]) / mu_w[-2])

        print_a = False
        if print_a:
                print('The CT number is %.3f HU for adipose.' % HU_adip)
                print(HU_adip_range)
                print('The CT number is %.3f HU for adipose for 120 kV.' % HU_adip)
                print(HU_adip)

        plot_blood = False
        if plot_blood:
                plt.plot(hv, psi, label='Original psi')
                plt.plot(hv, mu_adip_v[0], label='Filtered through Al')
                plt.plot(hv, mu_adip_v[1], label='Both filters')
                plt.plot(hv, mu_adip_v[-1], label='Only bodyfilter')
                plt.legend()
```

```
            plt.title('see how t - Al - adipose')
            plt.xlabel('energy [hv]')
            plt.ylabel('$ \psi $')
            plt.show()

##########################################################################
#####
##########################################################################
#####
##########################################################################
#####
#####
                                    #####
#####
                                    #####
#####                         IMPLEMENT THE STOPPING POWER RATIO
          #####
#####
                                    #####
#####
                                    #####
##########################################################################
#####
##########################################################################
#####
##########################################################################
#####

# Now I will implement the Stopping Power by using the Bethe Block equation
# where I will find beta from the kinetic energy and by tabulated values for
# Z/A and I. I will still look at soft tissue, lung and bone. I will set the
# energy range frome 1 MeV to 250 MeV. s

T_en = np.arange(1e3, 251e3, 1e3)                    # Kinetic energy [keV]
M0_c = 938.28 * 1e3                                  # Rest energy [keV]
z       = 1                                          #
Charge of proton

# print(T_en)

Z_A_cortical =      0.51478
I_cortical      =   112.0
Z_A_soft        =   0.54996
I_soft          =   74.7
Z_A_lung        =   0.55048
I_lung          =   75.2
Z_A_w           =       0.55508
I_w             =   75.0
Z_A_brain       =   0.55239
I_brain         =   73.9
```

```
Z_A_blood     =     0.54999
I_blood       =     75.2
Z_A_poly      =     0.57033
I_poly        =     57.4
Z_A_adip      =     0.55579
I_adip        =     64.8

# A function to calculate the beta value, dependent on the kinetic energy (T) of
# the proton. Thie reutrn will be unitless
def beta(T):
        b = ( 1 - ( 1 / ( (T/M0_c) + 1  ) )**2 )**(1./2.)

        return b

# A function to calculate the mass stopping power, takes the kinetic energy (T),
# mean excitation energy (I) and atomic number-to-mass (Z/A) as input variables
# the return will have the units [MeV/(g/cm^2)]
# a function of proton energy for given material
def dedx(T, I, Z_A):
        be = beta(T)

        # Stopping power with units [MeV/(g/cm^2)]
        SP = 0.3071 * Z_A * z**2 * (1/be**2) * (13.8373 + np.log( be**2 / \
                ( 1- be**2)) - be**2 - np.log(I) )

        return SP

#########################################################################
#####
#########################################################################
#####
#########################################################################
#####
#####
                                        #####
#####           FIND THE MASS STOPPING POWER FOR DIFFERENT MEDIUMS
                  #####
#####
                                        #####
#########################################################################
#####
#########################################################################
#####
#########################################################################
#####

# Want to calculate the different Mass stopping power
SP_w = True
if SP_w:
        sp_w = dedx(T_en, I_w, Z_A_w)
```

```
SP_Cort = True
if SP_Cort:
        sp_cort = dedx(T_en, I_cortical, Z_A_cortical)

SP_soft = True
if SP_soft:
        sp_soft = dedx(T_en, I_soft, Z_A_soft)

SP_lung = True
if SP_lung:
        sp_lung = dedx(T_en, I_lung, Z_A_lung)

SP_brain = True
if SP_brain:
        sp_brain = dedx(T_en, I_brain, Z_A_brain)

SP_blood = True
if SP_blood:
        sp_blood = dedx(T_en, I_blood, Z_A_blood)

SP_poly = True
if SP_poly:
        sp_poly = dedx(T_en, I_poly, Z_A_poly)

SP_adip = True
if SP_adip:
        sp_adip = dedx(T_en, I_adip, Z_A_adip)

#############################################################################
#####
#####                                P L O T  C H E C K (ATTIX)
                        #####
#############################################################################
#####

attix_check_plot = False
if attix_check_plot:
        plt.plot(T_en/M0_c, sp_w,              label='Water')
        plt.plot(T_en/M0_c, sp_lung,           label='Lung')
        plt.plot(T_en/M0_c, sp_soft, label='Soft tissue')
        plt.plot(T_en/M0_c, sp_cort,           label='Cotrical bone')
        plt.plot(T_en/M0_c, sp_brain,          label='Brain')
        plt.plot(T_en/M0_c, sp_blood,          label='Blood')
        plt.plot(T_en/M0_c, sp_poly,           label='Polyethylene')
        plt.plot(T_en/M0_c, sp_adip,           label='Adipose')
        plt.xscale('log')
        plt.legend()
        plt.title('Mass stopping power')
        plt.xlabel('$T/M_0 c^2$')
```

```
        plt.ylabel('$ dT / \\rho dx $')
        plt.show()

# Check the Mass Stopping power ratio vs HU unit, single energy setting
massSPR = False
if massSPR:

        # Find the  mean mass SP ratio
        sp_lung_ratio  = np.mean(sp_lung    / sp_w)
        sp_soft_ratio  = np.mean(sp_soft    / sp_w)
        sp_cort_ratio  = np.mean(sp_cort    / sp_w)
        sp_brain_ratio         = np.mean(sp_brain   / sp_w)
        sp_blood_ratio         = np.mean(sp_blood  / sp_w)
        sp_poly_ratio  = np.mean(sp_poly    / sp_w)
        sp_adip_ratio  = np.mean(sp_adip    / sp_w)

        # Make a 2d array with the Hounsfield unit (HU) and the mean mass SP ratio
        Lung_uni_mass       = np.array((HU_al_lung, sp_lung_ratio))
        Cort_uni_mass       = np.array((HU_cort, sp_cort_ratio))
        Soft_uni_mass       = np.array((HU,              sp_soft_ratio))
        Brain_uni_mass      = np.array((HU_brain,        sp_brain_ratio))
        Blood_uni_mass      = np.array((HU_blood,        sp_blood_ratio))
        Poly_uni_mass       = np.array((HU_poly,         sp_poly_ratio))
        Adip_uni_mass       = np.array((HU_adip,         sp_adip_ratio))

        #print(Lung_uni[:], Cort_uni[:], Soft_uni[:])

        # Makes a plot with HU vs massSPR
        plott = False
        if plott:
                plt.plot(Lung_uni_mass[0],   Lung_uni_mass[1],    'o', label='Lung')
                plt.plot(Cort_uni_mass[0],   Cort_uni_mass[1],    'o', label='Cortical bone')
                plt.plot(Soft_uni_mass[0],   Soft_uni_mass[1],    'o', label='Soft Tissue')
                plt.plot(Brain_uni_mass[0], Brain_uni_mass[1],    'o', label='Brain')
                plt.plot(Blood_uni_mass[0], Blood_uni_mass[1],    'o', label='Blood')
                plt.plot(Poly_uni_mass[0],   Poly_uni_mass[1],    'o', label='Polyethylene')
                plt.plot(Adip_uni_mass[0],   Adip_uni_mass[1],    'o', label='Adipose')
                plt.legend()
                plt.title('Mass stopping power ratio vs HU')
                plt.xlabel('HU')
                plt.ylabel('Mass stopping power ratio')
                plt.show()

# Stopping power, not mass stopping power ratio, single energy setting
# water is still in mass stopping power
SPR = True
if SPR:

        # Calculate the mean SP ratio
        sp_lung_r       = np.mean( ( sp_lung  * Lung_dens)          /(sp_w * Water_dens) )
```

```
        sp_soft_r       = np.mean( ( sp_soft  * Soft_t_dens) /(sp_w * Water_dens) )
        sp_cort_r       = np.mean( ( sp_cort  * Cort_dens)   /(sp_w * Water_dens) )
        sp_brain_r      = np.mean( ( sp_brain * Brain_dens)  /(sp_w * Water_dens) )
        sp_blood_r      = np.mean( ( sp_blood * Blood_dens)  /(sp_w * Water_dens) )
        sp_poly_r       = np.mean( ( sp_poly  * Poly_dens)   /(sp_w * Water_dens) )
        sp_adip_r       = np.mean( ( sp_adip  * Adip_dens)   /(sp_w * Water_dens) )

        # Make a 2d array with HU and meanSPR
        Lung_uni        = np.array((HU_al_lung, sp_lung_r))
        Cort_uni        = np.array((HU_cort, sp_cort_r))
        Soft_uni        = np.array((HU,              sp_soft_r))
        Brain_uni       = np.array((HU_brain,        sp_brain_r))
        Blood_uni       = np.array((HU_blood,        sp_blood_r))
        Poly_uni        = np.array((HU_poly,         sp_poly_r))
        Adip_uni        = np.array((HU_adip,         sp_adip_r))

        # Make a plot with HU vs SPR
        plott = False
        if plott:
                plt.plot(Lung_uni[0], Lung_uni[1], 'o', label='Lung')
                plt.plot(Cort_uni[0], Cort_uni[1], 'o', label='Cortical bone')
                plt.plot(Soft_uni[0], Soft_uni[1], 'o', label='Soft Tissue')
                plt.plot(Brain_uni[0],Brain_uni[1],'o', label='Brain')
                plt.plot(Blood_uni[0],Blood_uni[1],'o', label='Blood')
                plt.plot(Poly_uni[0], Poly_uni[1], 'o', label='Polyethylene')
                plt.plot(Adip_uni[0], Adip_uni[1], 'o', label='Adipose')
                plt.legend(loc=4, prop={"size":10})
                plt.title('120 kV', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

        #print(Lung_uni_notmass[:], Cort_uni_notmass[:], Soft_uni_notmass[:])

# Energy spectra from 20keV to 140 keV for Stopping power ratio, not mass
SPR_range = True
if SPR_range:

        # Array with zeros, make an array with the SP ratio
        sp_lung_r_mono      = np.zeros(13)
        sp_soft_r_mono      = np.zeros(13)
        sp_cort_r_mono      = np.zeros(13)
        sp_brain_r_mono = np.zeros(13)
        sp_blood_r_mono = np.zeros(13)
        sp_poly_r_mono      = np.zeros(13)
        sp_adip_r_mono      = np.zeros(13)

        # For loop to insert the mean SP ratio
        for i in range(13):
```

```
            sp_lung_r_mono[i] = np.mean((sp_lung * Lung_dens)  /(sp_w * Water_dens))
            sp_soft_r_mono[i] = np.mean((sp_soft * Soft_t_dens)/(sp_w * Water_dens))
            sp_cort_r_mono[i] = np.mean((sp_cort * Cort_dens)  /(sp_w * Water_dens))
            sp_brain_r_mono[i]= np.mean((sp_brain* Brain_dens) /(sp_w * Water_dens))
            sp_blood_r_mono[i]= np.mean((sp_blood* Blood_dens) /(sp_w *
Water_dens))
            sp_poly_r_mono[i] = np.mean((sp_poly * Poly_dens)  /(sp_w * Water_dens))
            sp_adip_r_mono[i] = np.mean((sp_adip * Adip_dens)  /(sp_w * Water_dens))

            print(sp_blood_r_mono)

    # Make a 2d arrat with HU and SP ratio
    Lung_uni_m   = np.array((HU_lung,                sp_lung_r_mono))
    Cort_uni_m   = np.array((HU_cort_list,     sp_cort_r_mono))
    Soft_uni_m   = np.array((HU_soft_2,              sp_soft_r_mono))
    Brain_uni_m  = np.array((HU_brain_range, sp_brain_r_mono))
    Blood_uni_m  = np.array((HU_blood_range, sp_blood_r_mono))
    Poly_uni_m   = np.array((HU_poly_range, sp_poly_r_mono))
    Adip_uni_m   = np.array((HU_adip_range, sp_adip_r_mono))

    plot_spr_no_lines = False
    if plot_spr_no_lines:
            # Make a plot with the HU range and mass SP ratio range
            plt.plot(Lung_uni_m[0],      Lung_uni_m[1], 'o', label='Lung')
            plt.plot(Cort_uni_m[0],      Cort_uni_m[1], 'o', label='Cortical bone')
            plt.plot(Soft_uni_m[0],      Soft_uni_m[1], 'o', label='Soft Tissue' )
            plt.plot(Brain_uni_m[0],     Brain_uni_m[1],'o', label='Brain')
            plt.plot(Blood_uni_m[0],     Blood_uni_m[1],'o', label='Blood')
            plt.plot(Poly_uni_m[0],      Poly_uni_m[1], 'o', label='Polyethylene')
            plt.plot(Adip_uni_m[0],      Adip_uni_m[1], 'o', label='Adipose')
            plt.legend(loc=4, prop={"size":10})
            plt.title('Monoenergeitc', fontsize=14)
            plt.xlabel('HU', fontsize = 12)
            plt.ylabel('Stopping power ratio', fontsize=12)
            plt.grid(True, color='gainsboro')
            plt.show()

    # arrays filled with zeros
    x_en = np.zeros((11,7))
    y_en = np.zeros((11,7))

    # fill the arrays with energy from 40 to 140
    for i in range(11):
            x_en[i] = np.array((Lung_uni_m[0][i+2], Poly_uni_m[0][i+2], \
                            Adip_uni_m[0][i+2], Brain_uni_m[0][i+2], Soft_uni_m[0][i+2],
\
                            Blood_uni_m[0][i+2], Cort_uni_m[0][i+2]))

            y_en[i] = np.array((Lung_uni_m[1][i+2],  Poly_uni_m[1][i+2], \
```

```
                                Adip_uni_m[1][i+2],  Brain_uni_m[1][i+2],
Soft_uni_m[1][i+2],  \
                                Blood_uni_m[1][i+2], Cort_uni_m[1][i+2]))

        # make different types of plots
        plot_spr_lines = False
        if plot_spr_lines:
                plott_Taran = True
                if plott_Taran:
                        plt.plot(x_en[0], y_en[0],  '-o', label='40 keV', color = 'r')
                        #plt.plot(x_en[1], y_en[1],  '-o', label='50 keV', color ='c')
                        #plt.plot(x_en[2], y_en[2],  '-o', label='60 keV', color ='m')
                        #plt.plot(x_en[3], y_en[3],  '-o', label='70 keV', color = 'y')
                        #plt.plot(x_en[4], y_en[4],  '-o', label='80 keV', color = 'k')
                        #plt.plot(x_en[5], y_en[5],  '-o', label='90 keV', color = 'C0' )
                        #plt.plot(x_en[6], y_en[6],'-o', label='100 keV', color = 'tab:pink')
                        #plt.plot(x_en[7], y_en[7],'-o', label='110 keV', color = 'tab:cyan')
                        #plt.plot(x_en[8], y_en[8], '-o', label='120 keV',color ='tab:olive')
                        #plt.plot(x_en[9], y_en[9], '-o', label='130 keV',color ='gray')
                        plt.plot(x_en[10], y_en[10], '-o',label='140 keV',color ='orange')
                        plt.legend(loc=4, prop={"size":10})
                        plt.title('Monoenergeitc', fontsize=14)
                        plt.xlabel('HU', fontsize=12)
                        plt.ylabel('Stopping power ratio', fontsize=12)
                        plt.grid(True, color='gainsboro')
                        plt.show()

                plott_Eirik = True
                if plott_Eirik:
                        plt.plot(x_en[0], y_en[0],   label='40 keV',  color = 'r')
                        #plt.plot(x_en[1], y_en[1],   label='50 keV',  color = 'c')
                        #plt.plot(x_en[2], y_en[2],   label='60 keV',  color = 'm')
                        #plt.plot(x_en[3], y_en[3],   label='70 keV',  color = 'y')
                        #plt.plot(x_en[4], y_en[4],   label='80 keV',  color = 'k')
                        #plt.plot(x_en[5], y_en[5],   label='90 keV',  color = 'C0' )
                        #plt.plot(x_en[6], y_en[6], label='100 keV',  color = 'tab:pink')
                        #plt.plot(x_en[7], y_en[7], label='110 keV',  color = 'tab:cyan')
                        #plt.plot(x_en[8], y_en[8], label='120 keV', color = 'tab:olive')
                        #plt.plot(x_en[9], y_en[9], label='130 keV', color = 'gray')
                        plt.plot(x_en[10], y_en[10], label='140 keV', color = 'orange')
                        plt.legend(loc=4, prop={"size":10})
                        plt.title('Monoenergeitc', fontsize=14)
                        plt.xlabel('HU', fontsize=12)
                        plt.ylabel('Stopping power ratio', fontsize=12)
                        plt.grid(True, color='gainsboro')
                        plt.show()

#########################################################################
#####
```

```
#############################################################################
#####
#############################################################################
#####
#####
                                    #####
#####   I M P O R T  H U  F R O M  Q U A S A R  A N D  G A M M E X    #####
#####
                                    #####
#############################################################################
#####
#############################################################################
#####
#############################################################################
#####

import pandas as pd

def import_HU(path, SECT):
        # a function who take the path, and SECT = True or FALSE

        # open and read the excel file
        df = pd.read_excel(path)

        SECT = SECT
        if SECT:
                # HU from CT images

                #empty arrays
                CT_HU_1 = np.zeros(1)
                CT_HU_2 = np.zeros(1)
                CT_HU_3 = np.zeros(1)

                # append values to arrays
                CT_HU_1 = df.iloc[-1,1]
                CT_HU_2 = df.iloc[-1,2]
                CT_HU_3 = df.iloc[-1,3]

                #rename
                HU_1 = CT_HU_1
                HU_2 = CT_HU_2
                HU_3 = CT_HU_3

        else:
                # HU from DECT images

                # localise row and column
                b = df.iloc[:,0]
                N = len(b) - 1
                #print(N)
```

```
                # arrays with zeros
                HU_1 = np.zeros(N)
                HU_2 = np.zeros(N)
                HU_3 = np.zeros(N)

                # arrays with zeros
                for i in range(N):
                        HU_1[i] = df.iloc[i, 1]
                        HU_2[i] = df.iloc[i, 2]
                        HU_3[i] = df.iloc[i, 3]

        return HU_1, HU_2, HU_3

Impr_HU = True
if Impr_HU:
        # name of the wanted files
        loc1 = ["mu_gammex1.xlsx", "mu_gammex2.xlsx"]
        loc2 = ["mu_quasar1.xlsx", "mu_quasar2.xlsx"]

        # arrays filled with zeros
        Blood = np.zeros((2,11))
        Brain = np.zeros((2,11))
        Adipose = np.zeros((2,11))
        Lung = np.zeros((2,11))
        Bone = np.zeros((2,11))
        Poly = np.zeros((2,11))

        # arrays filled with zeros
        Blood_120 = np.zeros((2,1))
        Brain_120 = np.zeros((2,1))
        Adipose_120 = np.zeros((2,1))
        Lung_120 = np.zeros((2,1))
        Bone_120 = np.zeros((2,1))
        Poly_120 = np.zeros((2,1))

        for i in range(2):
                # Store the HU in different arrays
                # The first row is filled with backprojection data
                # and the second row is filled with the iterative data
                # Data from DECT
                Blood[i], Brain[i], Adipose[i] = import_HU(loc1[i], False)
                Lung[i], Bone[i], Poly[i] = import_HU(loc2[i], False)

                # Data from SECT
                Blood_120[i], Brain_120[i], Adipose_120[i] = import_HU(loc1[i], True)
                Lung_120[i], Bone_120[i], Poly_120[i] = import_HU(loc2[i], True)

################################################################################
#####
```

```
#############################################################################
#####
###
                                                    ###
###                                  H U (PHANTOM) -- S P R (PROGRAM)
                                ###
###
                                                ###
#############################################################################
#####
#############################################################################
#####

#calculate the SPR for plot with implemented data
# always set to True
cal_SPR_DECT = True
if cal_SPR_DECT:

        # Array with zeros, make an array with the SP ratio
        sp_lung_r_M  = np.zeros(11)
        sp_cort_r_M   = np.zeros(11)
        sp_brain_r_M          = np.zeros(11)
        sp_blood_r_M          = np.zeros(11)
        sp_poly_r_M  = np.zeros(11)
        sp_adip_r_M  = np.zeros(11)

        # For loop to insert the mean SP ratio
        for i in range(11):
                sp_lung_r_M[i] = np.mean((sp_lung * Lung_dens)  /(sp_w * Water_dens))
                sp_cort_r_M[i] = np.mean((sp_cort * Cort_dens)  /(sp_w * Water_dens))
                sp_brain_r_M[i]= np.mean((sp_brain * Brain_dens)/(sp_w * Water_dens))
                sp_blood_r_M[i]= np.mean((sp_blood * Blood_dens)/(sp_w * Water_dens))
                sp_poly_r_M[i] = np.mean((sp_poly * Poly_dens)  /(sp_w * Water_dens))
                sp_adip_r_M[i] = np.mean((sp_adip * Adip_dens)  /(sp_w * Water_dens))

        # Make a 2d arrat with HU and SP ratio
        Lung_uni_Mb          = np.array((Lung[0],  sp_lung_r_M))
        Cort_uni_Mb  = np.array((Bone[0],  sp_cort_r_M))
        Brain_uni_Mb         = np.array((Brain[0],  sp_brain_r_M))
        Blood_uni_Mb         = np.array((Blood[0],          sp_blood_r_M))
        Poly_uni_Mb  = np.array((Poly[0],   sp_poly_r_M))
        Adip_uni_Mb          = np.array((Adipose[0], sp_adip_r_M))

# stopping power ratio, HU from phantoms
# for DECT image - monoenergetic images
SPR_range_imp_back = True
if SPR_range_imp_back:

        # Make a 2d arrat with HU and SP ratio
        Lung_uni_Mb          = np.array((Lung[0],  sp_lung_r_M))
```

```
Cort_uni_Mb  = np.array((Bone[0],  sp_cort_r_M))
Brain_uni_Mb          = np.array((Brain[0],  sp_brain_r_M))
Blood_uni_Mb          = np.array((Blood[0],          sp_blood_r_M))
Poly_uni_Mb  = np.array((Poly[0],   sp_poly_r_M))
Adip_uni_Mb           = np.array((Adipose[0], sp_adip_r_M))

plot_spr_no_lines = False
if plot_spr_no_lines:
        # Make a plot with the HU range and mass SP ratio range
        plt.plot(Lung_uni_Mb[0], Lung_uni_Mb[1], 'o', label='Lung')
        plt.plot(Cort_uni_Mb[0], Cort_uni_Mb[1], 'o', label='Cortical bone')
        plt.plot(Brain_uni_Mb[0],Brain_uni_Mb[1],'o', label='Brain')
        plt.plot(Blood_uni_Mb[0],Blood_uni_Mb[1],'o', label='Blood')
        plt.plot(Poly_uni_Mb[0], Poly_uni_Mb[1], 'o', label='Polyethylene')
        plt.plot(Adip_uni_Mb[0], Adip_uni_Mb[1], 'o', label='Adipose')
        plt.legend(loc=4, prop={"size":10})
        plt.title('DECT', fontsize=14)
        plt.xlabel('HU', fontsize=12)
        plt.ylabel('Stopping power ratio', fontsize=12)
        plt.grid(True, color='gainsboro')
        plt.show()

x_en_b = np.zeros((11,6))
y_en_b = np.zeros((11,6))

for i in range(11):
        x_en_b[i] = np.array((Lung_uni_Mb[0][i], Poly_uni_Mb[0][i], \
                        Adip_uni_Mb[0][i], Brain_uni_Mb[0][i], \
                        Blood_uni_Mb[0][i], Cort_uni_Mb[0][i]))

        y_en_b[i] = np.array((Lung_uni_Mb[1][i], Poly_uni_Mb[1][i], \
                        Adip_uni_Mb[1][i],  Brain_uni_Mb[1][i],  \
                        Blood_uni_Mb[1][i], Cort_uni_Mb[1][i]))

plot_spr_lines = False
if plot_spr_lines:
        plott_Taran = True
        if plott_Taran:
                plt.plot(x_en_b[0], y_en_b[0],  '-o', label='40 keV', color = 'r')
                plt.plot(x_en_b[1], y_en_b[1],  '-o', label='50 keV', color ='c')
                plt.plot(x_en_b[2], y_en_b[2],  '-o', label='60 keV', color ='m')
                plt.plot(x_en_b[3], y_en_b[3],  '-o', label='70 keV', color = 'y')
                plt.plot(x_en_b[4], y_en_b[4],  '-o', label='80 keV', color = 'k')
                plt.plot(x_en_b[5], y_en_b[5],  '-o', label='90 keV', color = 'C0' )
                plt.plot(x_en_b[6], y_en_b[6],'-o', label='100 keV', color = 'tab:pink')
                plt.plot(x_en_b[7], y_en_b[7],'-o', label='110 keV', color = 'tab:cyan')
                plt.plot(x_en_b[8], y_en_b[8], '-o', label='120 keV',color ='tab:olive')
                plt.plot(x_en_b[9], y_en_b[9], '-o', label='130 keV',color ='gray')
                plt.plot(x_en_b[10], y_en_b[10], '-o',label='140 keV',color ='orange')
                plt.legend(loc=4, prop={"size":10})
```

```
                plt.title('DECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

        plott_Eirik = True
        if plott_Eirik:
                plt.plot(x_en_b[0], y_en_b[0],   label='40 keV',  color = 'r')
                plt.plot(x_en_b[1], y_en_b[1],   label='50 keV',  color = 'c')
                plt.plot(x_en_b[2], y_en_b[2],   label='60 keV',  color = 'm')
                plt.plot(x_en_b[3], y_en_b[3],   label='70 keV',  color = 'y')
                plt.plot(x_en_b[4], y_en_b[4],   label='80 keV',  color = 'k')
                plt.plot(x_en_b[5], y_en_b[5],   label='90 keV',  color = 'C0' )
                plt.plot(x_en_b[6], y_en_b[6], label='100 keV',  color = 'tab:pink')
                plt.plot(x_en_b[7], y_en_b[7], label='110 keV',  color = 'tab:cyan')
                plt.plot(x_en_b[8], y_en_b[8], label='120 keV', color = 'tab:olive')
                plt.plot(x_en_b[9], y_en_b[9], label='130 keV', color = 'gray')
                plt.plot(x_en_b[10], y_en_b[10], label='140 keV', color = 'orange')
                plt.legend(loc=4, prop={"size":10})
                plt.title('DECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

        plot_greier = False
        if plot_greier:
                plt.plot(x_40kev[1:5], y_40kev[1:5], '-o', label = '40 keV')
                plt.title('DECT')
                plt.xlabel('HU')
                plt.ylabel('Stopping power ratio')
                plt.show()

SPR_range_imp_ite = True
if SPR_range_imp_ite:

        # Make a 2d arrat with HU and SP ratio
        Lung_uni_Mi = np.array((Lung[1],  sp_lung_r_M))
        Cort_uni_Mi  = np.array((Bone[1],  sp_cort_r_M))
        Brain_uni_Mi          = np.array((Brain[1],  sp_brain_r_M))
        Blood_uni_Mi          = np.array((Blood[1],          sp_blood_r_M))
        Poly_uni_Mi  = np.array((Poly[1],   sp_poly_r_M))
        Adip_uni_Mi  = np.array((Adipose[1], sp_adip_r_M))

        plot_spr_no_lines = False
        if plot_spr_no_lines:
                # Make a plot with the HU range and mass SP ratio range
                plt.plot(Lung_uni_Mi[0], Lung_uni_Mi[1], 'o', label='Lung')
                plt.plot(Cort_uni_Mi[0], Cort_uni_Mi[1], 'o', label='Cortical bone')
```

```
        plt.plot(Brain_uni_Mi[0],Brain_uni_Mi[1],'o', label='Brain')
        plt.plot(Blood_uni_Mi[0],Blood_uni_Mi[1],'o', label='Blood')
        plt.plot(Poly_uni_Mi[0], Poly_uni_Mi[1], 'o', label='Polyethylene')
        plt.plot(Adip_uni_Mi[0], Adip_uni_Mi[1], 'o', label='Adipose')
        plt.legend(loc=4, prop={"size":10})
        plt.title('DECT', fontsize=14)
        plt.xlabel('HU', fontsize=12)
        plt.ylabel('Stopping power ratio', fontsize=12)
        plt.grid(True, color='gainsboro')
        plt.show()

x_en_i = np.zeros((11,6))
y_en_i = np.zeros((11,6))

for i in range(11):
        x_en_i[i] = np.array((Lung_uni_Mi[0][i], Poly_uni_Mi[0][i], \
                        Adip_uni_Mi[0][i], Brain_uni_Mi[0][i], \
                        Blood_uni_Mi[0][i], Cort_uni_Mi[0][i]))

        y_en_i[i] = np.array((Lung_uni_Mi[1][i], Poly_uni_Mi[1][i], \
                        Adip_uni_Mi[1][i],  Brain_uni_Mi[1][i],  \
                        Blood_uni_Mi[1][i], Cort_uni_Mi[1][i]))

plot_spr_lines = False
if plot_spr_lines:
        plott_Taran = True
        if plott_Taran:
                plt.plot(x_en_i[0], y_en_i[0],  '-o', label='40 keV', color = 'r')
                plt.plot(x_en_i[1], y_en_i[1],  '-o', label='50 keV', color ='c')
                plt.plot(x_en_i[2], y_en_i[2],  '-o', label='60 keV', color ='m')
                plt.plot(x_en_i[3], y_en_i[3],  '-o', label='70 keV', color = 'y')
                plt.plot(x_en_i[4], y_en_i[4],  '-o', label='80 keV', color = 'k')
                plt.plot(x_en_i[5], y_en_i[5],  '-o', label='90 keV', color = 'C0' )
                plt.plot(x_en_i[6], y_en_i[6],'-o', label='100 keV', color = 'tab:pink')
                plt.plot(x_en_i[7], y_en_i[7],'-o', label='110 keV', color = 'tab:cyan')
                plt.plot(x_en_i[8], y_en_i[8], '-o', label='120 keV',color ='tab:olive')
                plt.plot(x_en_i[9], y_en_i[9], '-o', label='130 keV',color ='gray')
                plt.plot(x_en_i[10], y_en_i[10], '-o',label='140 keV',color ='orange')
                plt.legend(loc=4, prop={"size":10})
                plt.title('DECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

        plott_Eirik = True
        if plott_Eirik:
                plt.plot(x_en_i[0], y_en_i[0],   label='40 keV',  color = 'r')
                plt.plot(x_en_i[1], y_en_i[1],   label='50 keV',  color = 'c')
                plt.plot(x_en_i[2], y_en_i[2],   label='60 keV',  color = 'm')
```

```
                    plt.plot(x_en_i[3], y_en_i[3],   label='70 keV',  color = 'y')
                    plt.plot(x_en_i[4], y_en_i[4],   label='80 keV',  color = 'k')
                    plt.plot(x_en_i[5], y_en_i[5],   label='90 keV',  color = 'C0' )
                    plt.plot(x_en_i[6], y_en_i[6], label='100 keV',  color = 'tab:pink')
                    plt.plot(x_en_i[7], y_en_i[7], label='110 keV',  color = 'tab:cyan')
                    plt.plot(x_en_i[8], y_en_i[8], label='120 keV', color = 'tab:olive')
                    plt.plot(x_en_i[9], y_en_i[9], label='130 keV', color = 'gray')
                    plt.plot(x_en_i[10], y_en_i[10], label='140 keV', color = 'orange')
                    plt.legend(loc=4, prop={"size":10})
                    plt.title('DECT', fontsize=14)
                    plt.xlabel('HU', fontsize=12)
                    plt.ylabel('Stopping power ratio', fontsize=12)
                    plt.grid(True, color='gainsboro')
                    plt.show()

            plot_greier = False
            if plot_greier:
                    plt.plot(x_40kev[1:5], y_40kev[1:5], '-o', label = '40 keV')
                    plt.title('DECT')
                    plt.xlabel('HU')
                    plt.ylabel('Stopping power ratio')
                    plt.show()


#Calculates the spr for use in plot for implemented data
# always True since it have to be used
cal_SPR_SECT = True
if cal_SPR_SECT:

        sp_lung_r_CT        = np.zeros(1)
        sp_cort_r_CT = np.zeros(1)
        sp_brain_r_CT       = np.zeros(1)
        sp_blood_r_CT       = np.zeros(1)
        sp_poly_r_CT        = np.zeros(1)
        sp_adip_r_CT        = np.zeros(1)

        # Calculate the mean SP ratio
        sp_lung_r_CT[0]     = np.mean(( sp_lung  * Lung_dens) / (sp_w * Water_dens))
        sp_cort_r_CT[0]     = np.mean(( sp_cort  * Cort_dens) / (sp_w * Water_dens))
        sp_brain_r_CT[0]    = np.mean(( sp_brain * Brain_dens)/ (sp_w * Water_dens))
        sp_blood_r_CT[0]    = np.mean(( sp_blood * Blood_dens)/ (sp_w * Water_dens))
        sp_poly_r_CT[0]     = np.mean(( sp_poly  * Poly_dens) / (sp_w * Water_dens))
        sp_adip_r_CT[0]     = np.mean(( sp_adip  * Adip_dens) / (sp_w * Water_dens))

# stopping power ratio vs HU for the implemented HU values
# for CT image
SPR_phantom_back = True
if SPR_phantom_back:

        # Make a 2d array with HU and meanSPR
```

```
        Lung_uni_CT         = np.array((Lung_120[0],          sp_lung_r_CT))
        Cort_uni_CT  = np.array((Bone_120[0],          sp_cort_r_CT))
        Brain_uni_CT        = np.array((Brain_120[0],         sp_brain_r_CT))
        Blood_uni_CT        = np.array((Blood_120[0],         sp_blood_r_CT))
        Poly_uni_CT  = np.array((Poly_120[0],          sp_poly_r_CT))
        Adip_uni_CT = np.array((Adipose_120[0], sp_adip_r_CT))

        # Make a plot with HU vs SPR
        plott = False
        if plott:
                plt.plot(Lung_uni_CT[0],  Lung_uni_CT[1], 'o', label='Lung')
                plt.plot(Cort_uni_CT[0],  Cort_uni_CT[1], 'o', label='Cortical bone')
                plt.plot(Brain_uni_CT[0], Brain_uni_CT[1],'o', label='Brain')
                plt.plot(Blood_uni_CT[0], Blood_uni_CT[1],'o', label='Blood')
                plt.plot(Poly_uni_CT[0],  Poly_uni_CT[1], 'o', label='Polyethylene')
                plt.plot(Adip_uni_CT[0],  Adip_uni_CT[1], 'o', label='Adipose')
                plt.legend(loc=4, prop={"size":10})
                plt.title('SECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

# SPR vs HU for CT image with iterative reconstruction
# implemented data
SPR_phantom_it = True
if SPR_phantom_it:

        # Make a 2d array with HU and meanSPR
        Lung_uni_CT_i       = np.array((Lung_120[1],          sp_lung_r_CT))
        Cort_uni_CT_i       = np.array((Bone_120[1],          sp_cort_r_CT))
        Brain_uni_CT_i      = np.array((Brain_120[1],         sp_brain_r_CT))
        Blood_uni_CT_i      = np.array((Blood_120[1],         sp_blood_r_CT))
        Poly_uni_CT_i       = np.array((Poly_120[1],          sp_poly_r_CT))
        Adip_uni_CT_i       = np.array((Adipose_120[1], sp_adip_r_CT))

        # Make a plot with HU vs SPR
        plott = False
        if plott:
                plt.plot(Lung_uni_CT_i[0],  Lung_uni_CT_i[1], 'o', label='Lung')
                plt.plot(Cort_uni_CT_i[0],  Cort_uni_CT_i[1], 'o', label='Cortical bone')
                plt.plot(Brain_uni_CT_i[0], Brain_uni_CT_i[1],'o', label='Brain')
                plt.plot(Blood_uni_CT_i[0], Blood_uni_CT_i[1],'o', label='Blood')
                plt.plot(Poly_uni_CT_i[0],  Poly_uni_CT_i[1], 'o', label='Polyethylene')
                plt.plot(Adip_uni_CT_i[0],  Adip_uni_CT_i[1], 'o', label='Adipose')
                plt.legend(loc=4, prop={"size":10})
                plt.title('SECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
```

```
                plt.show()

SECT_combined = True
if SECT_combined:
        plott = False
        if plott:
                plt.plot(Lung_uni_CT_i[0],  Lung_uni_CT_i[1], 'o', label='Lung')
                plt.plot(Cort_uni_CT_i[0],  Cort_uni_CT_i[1], 'o', label='Cortical bone')
                plt.plot(Brain_uni_CT_i[0], Brain_uni_CT_i[1],'o', label='Brain')
                plt.plot(Blood_uni_CT_i[0], Blood_uni_CT_i[1],'o', label='Blood')
                plt.plot(Poly_uni_CT_i[0],  Poly_uni_CT_i[1], 'o', label='Polyethylene')
                plt.plot(Adip_uni_CT_i[0],  Adip_uni_CT_i[1], 'o', label='Adipose')
                plt.plot(Lung_uni_CT[0],  Lung_uni_CT[1], '*')
                plt.plot(Cort_uni_CT[0],  Cort_uni_CT[1], '*')
                plt.plot(Brain_uni_CT[0], Brain_uni_CT[1],'*')
                plt.plot(Blood_uni_CT[0], Blood_uni_CT[1],'*')
                plt.plot(Poly_uni_CT[0],  Poly_uni_CT[1], '*')
                plt.plot(Adip_uni_CT[0],  Adip_uni_CT[1], '*')
                blue_line = mlines.Line2D([], [], marker='*',\
                  markersize=9, label='Backprojection')
                grey_line = mlines.Line2D([], [], marker='o',\
                  markersize=9, label='Iterative')
                plt.legend(handles=[blue_line, grey_line])
                plt.title('SECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()


DECT_combined = True
if DECT_combined:
        plott = False
        if plott:
                plt.plot(Cort_uni_Mi[0], Cort_uni_Mi[1], 'o', label='Cortical bone')
                plt.plot(Brain_uni_Mi[0],Brain_uni_Mi[1],'o', label='Brain')
                plt.plot(Blood_uni_Mi[0],Blood_uni_Mi[1],'o', label='Blood')
                plt.plot(Poly_uni_Mi[0], Poly_uni_Mi[1], 'o', label='Polyethylene')
                plt.plot(Adip_uni_Mi[0], Adip_uni_Mi[1], 'o', label='Adipose')
                plt.plot(Lung_uni_Mb[0], Lung_uni_Mb[1], '*', label='Lung')
                plt.plot(Cort_uni_Mb[0], Cort_uni_Mb[1], '*', label='Cortical bone')
                plt.plot(Brain_uni_Mb[0],Brain_uni_Mb[1],'*', label='Brain')
                plt.plot(Blood_uni_Mb[0],Blood_uni_Mb[1],'*', label='Blood')
                plt.plot(Poly_uni_Mb[0], Poly_uni_Mb[1], '*', label='Polyethylene')
                plt.plot(Adip_uni_Mb[0], Adip_uni_Mb[1], '*', label='Adipose')
                plt.plot(Lung_uni_Mi[0], Lung_uni_Mi[1], 'o', label='Lung')
                blue_line = mlines.Line2D([], [], marker='*',\
                  markersize=9, label='Backprojection')
                grey_line = mlines.Line2D([], [], marker='o',\
                  markersize=9, label='Iterative')
```

```
                plt.legend(handles=[blue_line, grey_line])
                plt.title('DECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()


combined_mono_back = True
if combined_mono_back:
        plott = False
        if plott:
                plt.plot(x_en_b[0],  y_en_b[0],  'o', label='40 keV M')
                plt.plot(x_en[2],          y_en[2],          'o', label='40 keV S')
                plt.plot(x_en_b[4], y_en_b[4], 'o', label='80 keV M')
                plt.plot(x_en[6],          y_en[6],          'o', label='80 keV S')
                plt.plot(x_en_b[-1], y_en_b[-1], 'o', label='140 keV M')
                plt.plot(x_en[-1],         y_en[-1],         'o', label='140 keV S')
                plt.plot(Brain_uni_CT[0], Brain_uni_CT[1], 'o', label='CT_Brain' )
                plt.plot(Blood_uni_CT[0], Blood_uni_CT[1], 'o', label='CT_Blood' )
                plt.plot(Adip_uni_CT[0], Adip_uni_CT[1], 'o', label='CT_Adip' )
                plt.plot(Poly_uni_CT[0], Poly_uni_CT[1], 'o', label='CT_Poly' )
                plt.legend(loc=4, prop={"size":10})
                plt.title('Combined', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

        plott_zoom = False
        if plott_zoom:
                fig, ax = plt.subplots()

                ax.plot(x_en_b[0],  y_en_b[0],  'o', label='40 keV M')
                ax.plot(x_en[2],          y_en[2],          'o', label='40 keV S')
                ax.plot(x_en_b[4], y_en_b[4], 'o', label='80 keV M')
                ax.plot(x_en[6],          y_en[6],          'o', label='80 keV S')
                ax.plot(x_en_b[-1], y_en_b[-1], 'o', label='140 keV M')
                ax.plot(x_en[-1],         y_en[-1],         'o', label='140 keV S')
                ax.plot(Brain_uni_CT[0], Brain_uni_CT[1], 'o', label='CT_Brain' )
                ax.plot(Blood_uni_CT[0], Blood_uni_CT[1], 'o', label='CT_Blood' )
                ax.plot(Adip_uni_CT[0], Adip_uni_CT[1], 'o', label='CT_Adip' )
                ax.plot(Poly_uni_CT[0], Poly_uni_CT[1], 'o', label='CT_Poly' )
                ax.grid(True, color='gainsboro')
                ax.legend(loc=4)

                axins = zoomed_inset_axes(ax, 5, loc=8) # zoom = 5

                axins.plot(x_en_b[0],  y_en_b[0], 'o')
                axins.plot(x_en[2],      y_en[2], 'o')
```

```
                axins.plot(x_en_b[4], y_en_b[4], 'o')
                axins.plot(x_en[6],      y_en[6], 'o')
                axins.plot(x_en_b[-1], y_en_b[-1], 'o')
                axins.plot(x_en[-1],     y_en[-1], 'o')
                axins.set_xlim(-250, 120) # Limit the region for zoom
                axins.set_ylim(0.95, 1.07)
                axins.grid(True, color='gainsboro')

                plt.xticks(visible=False)  # Not present ticks
                plt.yticks(visible=False)

                ## draw a bbox of the region of the inset axes in the parent axes and
                ## connecting lines between the bbox and the inset axes area
                mark_inset(ax, axins, loc1=2, loc2=4, fc="none", ec="0.5")

                plt.draw()
                ax.set_title('Combined')
                ax.set_xlabel('HU')
                ax.set_ylabel('Stopping power ratio')
                plt.show()

kV_combined = False
if kV_combined:
        plott = True
        if plott:
                plt.plot(Lung_uni[0], Lung_uni[1], 'o', color='tab:blue')
                plt.plot(Cort_uni[0], Cort_uni[1], 'o', color='tab:pink')
                plt.plot(Brain_uni[0],Brain_uni[1],'o', color='tab:red')
                plt.plot(Blood_uni[0],Blood_uni[1],'o', color='tab:green')
                plt.plot(Poly_uni[0], Poly_uni[1], 'o', color='tab:purple')
                plt.plot(Adip_uni[0], Adip_uni[1], 'o', color='tab:cyan')
                plt.plot(Lung_uni_CT[0],  Lung_uni_CT[1], '*', color='tab:blue')
                plt.plot(Cort_uni_CT[0],  Cort_uni_CT[1], '*', color='tab:pink')
                plt.plot(Brain_uni_CT[0], Brain_uni_CT[1],'*', color='tab:red')
                plt.plot(Blood_uni_CT[0], Blood_uni_CT[1],'*', color='tab:green')
                plt.plot(Poly_uni_CT[0],  Poly_uni_CT[1], '*', color='tab:purple')
                plt.plot(Adip_uni_CT[0],  Adip_uni_CT[1], '*', color='tab:cyan')
                blue_line = mlines.Line2D([], [], marker='*',\
                  markersize=9, label='Measured')
                grey_line = mlines.Line2D([], [], marker='o',\
                  markersize=9, label='Simulated')
                plt.legend(handles=[blue_line, grey_line])
                plt.title('SECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

# make a plot where simulated and measured values are combined
# kV values
```

```
combined_m_s = True
if combined_m_s:
        #reorganize the values into x and y for simulated values
        x_120kv_s = np.array((Lung_uni[0], Poly_uni[0], Adip_uni[0],\
                              Brain_uni[0], Blood_uni[0], Cort_uni[0]  ))
        y_120kv_s = np.array((Lung_uni[1], Poly_uni[1], Adip_uni[1],\
                              Brain_uni[1], Blood_uni[1], Cort_uni[1]  ))

        #reorganize the values into x and y for simulated values
        x_120kv_m = np.array((Lung_uni_CT[0], Poly_uni_CT[0], Adip_uni_CT[0],\
                              Brain_uni_CT[0], Blood_uni_CT[0], Cort_uni_CT[0]  ))
        y_120kv_m = np.array((Lung_uni_CT[1], Poly_uni_CT[1], Adip_uni_CT[1],\
                              Brain_uni_CT[1], Blood_uni_CT[1], Cort_uni_CT[1]  ))
        plott = False
        if plott:

                #make a regular plot
                plt.plot(x_120kv_s, y_120kv_s, '-o', label='Simulated', color='tab:pink')
                plt.plot(x_120kv_m, y_120kv_m, '-v', label='Measured', color='tab:blue')
                plt.legend(loc=4)
                plt.title('SECT', fontsize=14)
                plt.xlabel('HU', fontsize=12)
                plt.ylabel('Stopping power ratio', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

                # make a plot with zoomed area
                fig, ax = plt.subplots()

                ax.plot(x_120kv_s, y_120kv_s, '-o', color='tab:pink')
                ax.plot(x_120kv_m, y_120kv_m, '-v', color='tab:blue')

                blue_line = mlines.Line2D([], [], marker='v',\
                  markersize=9, label='Measured', color='tab:blue')
                grey_line = mlines.Line2D([], [], marker='o',\
                  markersize=9, label='Simulated', color='tab:pink')
                ax.legend(handles=[blue_line, grey_line])

                axins = zoomed_inset_axes(ax, 4, loc=4, borderpad = 2) # zoom = 5
                axins.plot(x_120kv_s, y_120kv_s, '-o', color='tab:pink')
                axins.plot(x_120kv_m, y_120kv_m, '-v', color='tab:blue')

                axins.set_xlim(-200, 120) # Limit the region for zoom
                axins.set_ylim(0.95, 1.07)
                axins.grid(True, color='gainsboro')

                plt.xticks(visible=True)  # Not present ticks
                plt.yticks(visible=True)

                ## draw a bbox of the region of the inset axes in the parent axes and
```

```
                ## connecting lines between the bbox and the inset axes area
                mark_inset(ax, axins, loc1=2, loc2=4, fc="none", ec="0.5")

                plt.draw()
                ax.set_title('Combined kV', fontsize = 14)
                ax.set_xlabel('HU', fontsize=12)
                ax.set_ylabel('Stopping power ratio', fontsize=12)
                ax.grid(True, color='gainsboro')
                plt.show()

# monoenergetic difference between measured and simulated
difference_m_s_mono = False
if difference_m_s_mono:

        HU_diff_infl  = np.zeros(11)
        HU_diff_poly  = np.zeros(11)
        HU_diff_adip  = np.zeros(11)
        HU_diff_brain = np.zeros(11)
        HU_diff_blood = np.zeros(11)
        HU_diff_cort  = np.zeros(11)

        medium = ['Inf. lung','Polyethylene', 'Adipose',  'Brain', 'Blood', 'Cort. bone']

        # remove soft tissue values
        x_en_ny = np.delete(x_en, np.s_[4], axis=1)

        # Caluculate the difference between measured and simulated HU
        for i in range(11):
                # interested in difference, not size
                HU_diff_infl[i]  = abs(x_en_b[i][0] - x_en_ny[i][0])
                HU_diff_poly[i]  = abs(x_en_b[i][1] - x_en_ny[i][1])
                HU_diff_adip[i]  = abs(x_en_b[i][2] - x_en_ny[i][2])
                HU_diff_brain[i] = abs(x_en_b[i][3] - x_en_ny[i][3])
                HU_diff_blood[i] = abs(x_en_b[i][4] - x_en_ny[i][4])
                HU_diff_cort[i]  = abs(x_en_b[i][5] - x_en_ny[i][5])

        #fig,axes = plt.subplots()
        plott = True
        if plott:
                x_ny = np.array([1,2,3,4,5,6])
                fig, ax = plt.subplots()
                for i in range(11):
                        #make a plot with the differences
                        plt.plot(1, HU_diff_infl[i],  'o', color='tab:blue')
                        plt.plot(2, HU_diff_poly[i],  'o', color='tab:purple')
                        plt.plot(3, HU_diff_adip[i],  'o', color='tab:cyan')
                        plt.plot(4, HU_diff_brain[i], 'o', color='tab:red')
                        plt.plot(5, HU_diff_blood[i], 'o', color='tab:green')
                        plt.plot(6, HU_diff_cort[i],  'o', color='tab:pink')
```

```
                plt.xticks(x_ny, medium)
                plt.title('')
                plt.xlabel('Medium', fontsize='12')
                plt.ylabel('HU difference [$HU_m - HU_s]$', fontsize=12)
                plt.grid(True, color='gainsboro')
                plt.show()

# SPRV vs HU for 120 kV and 80 keV
# both measured
close_HU_combined = False
if close_HU_combined:

        x_80keV = x_en_b[4]
        y_80keV = y_en_b[4]

                # restore after increasing HU
        x_80keV = np.sort(x_80keV)
        y_80keV = np.sort(y_80keV)
        plott = False
        if plott:

                # new array with only 80 keV measured values
                x_80keV = x_en_b[4]
                y_80keV = y_en_b[4]

                # restore after increasing HU
                x_80keV = np.sort(x_80keV)
                y_80keV = np.sort(y_80keV)

                print(x_80keV[1])
                print(x_120kv_m[2])

                plott = False
                if plott:
                        fig, ax = plt.subplots()

                        ax.plot(x_80keV, y_80keV,    '-o', label='80 keV', color='tab:pink')
                        ax.plot(x_120kv_m, y_120kv_m,'-o', label='120 kV', color='tab:blue')
                        ax.legend()

                        ax.grid(True, color='gainsboro')

                        axins = zoomed_inset_axes(ax, 4, loc=4, borderpad=2) # zoom = 4

                        axins.plot(x_80keV, y_80keV,          '-o', color='tab:pink')
                        axins.plot(x_120kv_m, y_120kv_m, '-o', color='tab:blue')
                        axins.set_xlim(-100, 120) # Limit the region for zoom
                        axins.set_ylim(0.95, 1.07)
                        axins.grid(True, color='gainsboro')
```

```
                    #axins.yaxis.get_major_locator().set_params(nbins=7)
                    #axins.xaxis.get_major_locator().set_params(nbins=7)
                    plt.xticks(visible=True)  # Not present ticks
                    plt.yticks(visible=True)

                    ## draw a bbox of the region of the inset axes in the parent axes and
                    ## connecting lines between the bbox and the inset axes area
                    mark_inset(ax, axins, loc1=2, loc2=4, fc="none", ec="0.5")

                    plt.draw()
                    ax.set_title('')
                    ax.set_xlabel('HU', fontsize=12)
                    ax.set_ylabel('Stopping power ratio', fontsize=12)
                    plt.show()

close_HU_combined_more = True
if close_HU_combined_more:

        x_40keV = x_en_b[0]
        y_40keV = y_en_b[0]
        x_80keV = x_en_b[4]
        y_80keV = y_en_b[4]
        x_120keV = x_en_b[8]
        y_120keV = y_en_b[8]
        x_140keV = x_en_b[-1]
        y_140keV = y_en_b[-1]



        # restore after increasing HU
        x_80keV = np.sort(x_80keV)
        y_80keV = np.sort(y_80keV)
        x_120keV = np.sort(x_120keV)
        y_120keV = np.sort(y_120keV)
        x_140keV = np.sort(x_140keV)
        y_140keV = np.sort(y_140keV)


        plt.plot(x_120kv_m, y_120kv_m, '-o', label='120 kVp', color='tab:blue')
        plt.legend()

        #plt.xticks(x_ny, medium)
        plt.title('')
        plt.xlabel('HU', fontsize='12')
        plt.ylabel('Stopping Power Ratio', fontsize=12)
        plt.grid(True, color='gainsboro')
        plt.show()

        plott = False
        if plott:
```

```
plott = True
if plott:
        fig, ax = plt.subplots()

        ax.plot(x_80keV, y_80keV,    '-o', label='80 keV', color='tab:pink')
        ax.plot(x_120kv_m, y_120kv_m,'-o', label='120 kV', color='tab:blue')
        ax.plot(x_40keV, y_40keV, '-o', label='40 keV', color='tab:purple')
        ax.plot(x_140keV, y_140keV, '-o', label='140 keV', color='tab:red')
        ax.legend()
        #ax.title('SECT', fontsize=14)
        #ax.xlabel('HU', fontsize=12)
        #ax.ylabel('Stopping power ratio', fontsize=12)
        ax.grid(True, color='gainsboro')

        axins = zoomed_inset_axes(ax, 4, loc=4, borderpad=2) # zoom = 4

        axins.plot(x_80keV, y_80keV,          '-o', color='tab:pink')
        axins.plot(x_120kv_m, y_120kv_m, '-o', color='tab:blue')
        axins.plot(x_40keV, y_40keV, '-o',  color='tab:purple')
        axins.plot(x_140keV, y_140keV, '-o', color='tab:red')
        axins.set_xlim(-205, 120) # Limit the region for zoom
        axins.set_ylim(0.95, 1.07)
        axins.grid(True, color='gainsboro')

        #axins.yaxis.get_major_locator().set_params(nbins=7)
        #axins.xaxis.get_major_locator().set_params(nbins=7)
        plt.xticks(visible=True)  # Not present ticks
        plt.yticks(visible=True)
        #
        ## draw a bbox of the region of the inset axes in the parent axes and
        ## connecting lines between the bbox and the inset axes area
        mark_inset(ax, axins, loc1=2, loc2=4, fc="none", ec="0.5")

        plt.draw()
        ax.set_title('')
        ax.set_xlabel('HU', fontsize=12)
        ax.set_ylabel('Stopping power ratio', fontsize=12)
        plt.show()
```

Python code creating a tablet for extracting HU values from Quasar and Gammex phantom. Every calculation is set to False as default.

```
import pydicom as dcm
import numpy as np
import matplotlib.pyplot as plt
import glob
from openpyxl import Workbook
#import scipy as sp
#import scipy.interpolate

# a function to read the DICOM files, and sort them in correct order
def load_scans(path):
        files = glob.glob(path+"*") #Søker etter CT-filer i katalog. Evt bruk "*"

        zdim = len(files) # Antall filer = antall snitt

        test = dcm.dcmread(files[0]).pixel_array #leser ut et vilkårlig bilde, xog
        xdim = len(test[0]) # antall x = antall pixler i x
        ydim = len(test[1]) # antall y = antall pixler i y
        #print(test.shape)

        images = np.empty([xdim, ydim, zdim])
        sli_loc = np.empty([zdim])
        for z in range(zdim):
            sli_loc[z] = dcm.dcmread(files[z]).SliceLocation
        # Leste inn slice location, ikke riktig rekkefølge

        index_sort = np.argsort(sli_loc)

        for z in range(zdim):
            index = index_sort[z] # Leser inn bildene sortert etter slice location
            image = dcm.dcmread(files[index]).pixel_array
            images[:, :, z]= image

            #images = images.astype(np.int16)

        #print(images)
        return images


# The DICOM images have an Rescale intercept of -1024, and a Rescale Slope of 1,
# so to find the Hounsfield Unit in the DICOM image we have to use:
# U = m * SV + b
# where U = HU, m = Rescale Slope, SV = given value in the voxel, and
# b = Rescale intercept
def get_pixels_hu(scans):
    ima = scans
```

```
    ima += np.int16(-1024)

    return np.array(ima, dtype=np.int16)

'''
# Make a histogram to check the HU
plt.hist(imgs_to_process.flatten(), bins=50, color='c')
plt.xlabel("Hounsfield Units (HU)")
plt.ylabel("Frequency")
plt.show()
'''

# Show every 7th DICOM file, to figure out which slize we want to use
def sample_stack(stack, rows=4, cols=4, start_with=1, show_every=7):
    fig,ax = plt.subplots(rows,cols,figsize=[10,10])
    for i in range(rows*cols):
        ind = start_with + i*show_every
        ax[int(i/rows),int(i % rows)].set_title('slice %d' % ind)
        ax[int(i/rows),int(i % rows)].imshow(stack[:,:, ind], cmap=plt.cm.bone)
        ax[int(i/rows),int(i % rows)].axis('off')
    plt.show()

# A template for the circular ROI's in the GAMMEX phantom, returns the indexes
# where the ciruclar ROI's are located. This template is for the GAMMEX phantom
def template_gammex():

    # number of pixels in the radius
    ra = 10

    # choose which slize we want to use for the template
    dummy = patient_pixel[:,:, 92]

    '''
    circ = np.empty([512, 512])

    x0 = 435
    y0 = 252

    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                circ[i  + y0, j  + x0] = 12

    ind_cir = np.where( circ == 12)
    #print(ind_cir)

    #for i in range()
```

```
dummy = patient_dicom[:,:, 92]

dummy[ind_cir] = 0

plt.figure()
plt.imshow(dummy)
plt.show()

print(np.mean(dummy[ind_cir]))
print(np.std(dummy[ind_cir]))
'''
#####################################################################

# find the index for the first circle
x1 = 133 #122
y1 = 131 #122

# make a two dimentional array with size 512x512 filled with zeros
circle1 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 1. (array)
            circle1[i + y1, j + x1] = 1

# collect the indexes where the array is 1 (our wanted circle)
ind_circle1 = np.where( circle1 == 1)

#####################################################################

# find the index for the secound circle
x2 = 380 #372
y2 = 129 #118

# make a two dimentional array with size 512x512 filled with zeros
circle2 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 2.  (array)
            circle2[i + y2, j + x2] = 2
# collect the indexes where the array is 1 (our wanted circle)
```

```
ind_circle2 = np.where( circle2 == 2)

#########################################################################

# find the index for the third circle
x3 = 189 #179
y3 = 190 #179

# make a two dimentional array with size 512x512 filled with zeros
circle3 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 3. (array)
            circle3[i + y3, j + x3] = 3
# collect the indexes where the array is 1 (our wanted circle)
ind_circle3 = np.where( circle3 == 3)

#########################################################################

# find the index for the fourth circle
x4 = 255
y4 = 160

# make a two dimentional array with size 512x512 filled with zeros
circle4 = np.zeros([512,512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 4 (array)
            circle4[i + y4, j + x4] = 4

ind_circle4 = np.where( circle4 == 4)

#########################################################################

# find the index for the fifth circle
x5 = 83 #74
y5 = 256 #246

# make a two dimentional array with size 512x512 filled with zeros
circle5 = np.zeros([512, 512])
```

```
# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 5 (array)
            circle5[i + y5, j + x5] = 5
# collect the indexes where the array is 1 (our wanted circle)
ind_circle5 = np.where( circle5 == 5)

#########################################################################

# find the index for the sixth circle
x6 = 353 #343
y6 = 250 #240

# make a two dimentional array with size 512x512 filled with zeros
circle6 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 6 (array)
            circle6[i + y6, j + x6] = 6
# collect the indexes where the array is 1 (our wanted circle)
ind_circle6 = np.where( circle6 == 6)

#########################################################################

# find the index for the seventh circle
x7 = 194 #184
y7 = 321 #312

# make a two dimentional array with size 512x512 filled with zeros
circle7 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 7. (array)
            circle7[i + y7, j + x7] = 7
# collect the indexes where the array is 1 (our wanted circle)
```

```
    ind_circle7 = np.where( circle7 == 7)

    ######################################################################

    # find the index for the eight circle
    x8 = 247 #184
    y8 = 47 #312

    # make a two dimentional array with size 512x512 filled with zeros
    circle8 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 8. (array)
                circle8[i + y8, j + x8] = 8
    # collect the indexes where the array is 1 (our wanted circle)
    ind_circle8 = np.where( circle8 == 8)


    # test to check that the circle is in the right spot.
    # Set to 'True' if you want to plot the image
    test_spot = False
    if test_spot:
        dummy[ind_circle8] = 1200

        plt.figure()
        plt.imshow(dummy, cmap=plt.cm.bone)
        plt.show()

        #print(np.mean(dummy[ind_circle1])) # can only be used if dummy = -1200
        #print(np.std(dummy[ind_circle1])) # can only be used if dummy = -1200

    return ind_circle1, ind_circle2, ind_circle3, ind_circle4, ind_circle5,\
     ind_circle6, ind_circle7, ind_circle8

# A template for the circular ROI's in the QUASAR phantom, returns the indexes
# where the ciruclar ROI's are located. This template is for the GAMMEX phantom
def template_quasar():
    # number of pixels in the radius (same for both phantoms)
    ra = 10

    # choose what slize we want to use as the template
    dummy = patient_pixel_q[:,:, 106]

    '''
    circ = np.empty([512, 512])
```

```
    x0 = 257
    y0 = 252

    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                circ[i  + y0, j  + x0] = 12

    ind_cir = np.where( circ == 12)
    #print(ind_cir)
    '''


##############################################################################
#

    # find the index for the first circle
    x1 = 257
    y1 = 204

    # make a two dimentional array with size 512x512 filled with zeros
    circle1 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 1. (array)
                circle1[i + y1, j + x1] = 1
    # collect the indexes where the array is 1 (our wanted circle)
    ind_circle1 = np.where( circle1 == 1)


##############################################################################
#
    # find the index for the secound circle
    x2 = 305
    y2 = 252

    # make a two dimentional array with size 512x512 filled with zeros
    circle2 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
```

```
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 2. (array)
            circle2[i + y2, j + x2] = 2
    # collect the indexes where the array is 2 (our wanted circle)
    ind_circle2 = np.where( circle2 == 2)


#############################################################################
#
    # find the index for the third circle
    x3 = 257
    y3 = 300

    # make a two dimentional array with size 512x512 filled with zeros
    circle3 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 3. (array)
                circle3[i + y3, j + x3] = 3
    # collect the indexes where the array is 3 (our wanted circle)
    ind_circle3 = np.where( circle3 == 3)


#############################################################################
#
    # find the index for the fourth circle
    x4 = 210
    y4 = 252

    # make a two dimentional array with size 512x512 filled with zeros
    circle4 = np.zeros([512,512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 4. (array)
                circle4[i + y4, j + x4] = 4

    # collect the indexes where the array is 4 (our wanted circle)
    ind_circle4 = np.where( circle4 == 4)
```

```
#############################################################################
#
   # find the index for the fifth circle
   x5 = 257
   y5 = 252

   # make a two dimentional array with size 512x512 filled with zeros
   circle5 = np.zeros([512, 512])

   # creat mask in a circular pathern around the centrum of a choosen index
   for i in range(-ra, ra+1):
      for j in range(-ra, ra+1):
         r = np.sqrt( (i**2.0 ) + (j**2.0) )

         if r <= ra:
            #fill the indexes that is a part of the circle with 5. (array)
            circle5[i + y5, j + x5] = 5

   # collect the indexes where the array is 5 (our wanted circle)
   ind_circle5 = np.where( circle5 == 5)


#############################################################################
#

   # find the index for the sixth circle
   x6 = 257
   y6 = 160

   # make a two dimentional array with size 512x512 filled with zeros
   circle6 = np.zeros([512, 512])

   # creat mask in a circular pathern around the centrum of a choosen index
   for i in range(-ra, ra+1):
      for j in range(-ra, ra+1):
         r = np.sqrt( (i**2.0 ) + (j**2.0) )

         if r <= ra:
            #fill the indexes that is a part of the circle with 1. (array)
            circle6[i + y6, j + x6] = 6
   # collect the indexes where the array is 1 (our wanted circle)
   ind_circle6 = np.where( circle6 == 6)

   # test to check that the circle is in the right spot.
   # Set to 'True' if you want to plot the image
   check_temp = True
   if check_temp:
```

```
        dummy = patient_pixel_q[:,:, 106]

        dummy[ind_circle6] = 1200

        plt.figure(1)
        plt.imshow(dummy)
        plt.show()

    #print(np.mean(dummy[ind_cir]))
    #print(np.std(dummy[ind_cir]))

    return ind_circle1, ind_circle2, ind_circle3, ind_circle4, ind_circle5,\
     ind_circle6

# Able to save the data to a excel file
book = Workbook()

# Activate a sheet in the Workbook, able to write into it
spreadsheet = book.active

# Load the scans for the template
#############################################################################
#####
#####                G A M M E X   P H A N T O M                    #####
#############################################################################
#####

# path1 is a path to a Gammex phantom
path1="//Users/Sandra/Documents/UIO/Master/Masteroppgave/Method/CT-GE/\
PixPadZero/00000A7D Gammex single energi 120 kV AV50//"
# Katalog der CT-bildene ligger (NB ingen andre filer bør ligge der)

# start with the gammex phantom, loading the DICOM images from the map
patient_dicom = load_scans(path1)
# correctiong the pixel value to HU
patient_pixel = get_pixels_hu(patient_dicom)

# Check the stack, to find an image we can center around
imgs_to_process = patient_pixel

check_stack = False
if check_stack:
    sample_stack(imgs_to_process)

# Find the index for the ROI location for the different materials.
# A template to use on the DICOM images.
# Set as global variables
ind_blood70, ind_blood40, ind_brain, ind_water, ind_blood, ind_adipose, \
ind_air, ind_cont_g = template_gammex()
```

```
# From the stack image we know that the gammex has nice images around slize 92.
# Will use 5 slizes around number 92

#############################################################################
#####
#####              Q U A S A R   P H A N T O M                  #####
#############################################################################
#####

#path2 is a path to a quasar phantom
Path2 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000AD20 70 keV AV50 Quasar//"
#Path3 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//\
#CT-GE//PixPadZero//0000ECE8 Quasar single energi 120 kV AV50//"

# Start by loading the DICOM images from path2
patient_dicom_q = load_scans(Path2)
patient_pixel_q = get_pixels_hu(patient_dicom_q)

# Check the stack, to find an image we can center around
check_stack_q = False
if check_stack_q:
    sample_stack(patient_pixel_q)

# From stack check the interesting inserts looks best around slize 106.
# Will use 5 slizes around number 106
# Can alsow see that for the SECT we have to choose around slize 99 not 106
# But the location of the ROI's is still the same

# Find the index for the ROI location for the different materials.
# A template to use on the DICOM images.
# set as global variables
ind_lung, ind_dbone, ind_w, ind_inbone, ind_poly, ind_cont_q = template_quasar()

#############################################################################
#####
#############################################################################
#####
#####                                          #####
#####          FIND THE MEAN HU AND STD IN THE ROIS          #####
#####                                          #####
#############################################################################
#####
#############################################################################
#####

def Gammex_DECT_HU_SD(path):
    '''A function to calculate the Hounsfield Unit and the standard daviation in
    ROI's. This function will be used for the Gammex phantom only.'''
```

```
# Call the DICOM images
patient_dicom = load_scans(path)
# Correct the pixel value
patient_pixel = get_pixels_hu(patient_dicom)

# Number of coloums needed in the array
len_slize = len(patient_pixel[ind_blood70])

# Arrays filled with zeros
blood70     =  np.zeros((5,len_slize))
blood40     =  np.zeros((5,len_slize))
brain       =  np.zeros((5,len_slize))
water       =  np.zeros((5,len_slize))
blood       =  np.zeros((5,len_slize))
adipose     =  np.zeros((5,len_slize))
air         =  np.zeros((5,len_slize))
contr_g     =  np.zeros((5,len_slize))

# Only want to use 5 slizes around z = 92
for i in range(-2, 2+1):
   z = 92 + i
   image_slize = patient_pixel[:,:,z]

   # Store all the values in an 2d array, find the SD of the whole array
   blood70[i + 2, :]    =  image_slize[ind_blood70]
   blood40[i + 2, :]    =  image_slize[ind_blood40]
   brain[i   + 2, :]    =  image_slize[ind_brain]
   water[i   + 2, :]    =  image_slize[ind_water]
   blood[i   + 2, :]    =  image_slize[ind_blood]
   adipose[i + 2, :]    =  image_slize[ind_adipose]
   air[i     + 2, :]    =  image_slize[ind_air]
   contr_g[i   + 2, :]  =  image_slize[ind_cont_g]

# Calculate the mean HU for different material
mean_blood70 = np.mean(blood70)
mean_blood40 = np.mean(blood40)
mean_brain   = np.mean(brain)
mean_blood   = np.mean(blood)
mean_water   = np.mean(water)
mean_adipose = np.mean(adipose)
mean_air     = np.mean(air)
mean_cont_g  = np.mean(contr_g)

# Caclculate the standard daviation for the different material
sd_blood70  = np.std(blood70, ddof=1)
sd_blood40  = np.std(blood40, ddof=1)
sd_brain    = np.std(brain, ddof=1)
sd_blood    = np.std(blood, ddof=1)
sd_water    = np.std(water, ddof=1)
sd_adipose  = np.std(adipose, ddof=1)
```

```
    sd_air      = np.std(air, ddof=1)
    sd_cont_g   = np.std(contr_g, ddof=1)

    return mean_blood70, mean_blood40, mean_brain, mean_water, mean_blood, \
    mean_adipose, mean_air, sd_blood70, sd_blood40, sd_brain, sd_blood, sd_water,
sd_adipose, \
     sd_air, mean_cont_g, sd_cont_g


def QUASAR_DECT_HU_SD(path, CT):
    '''A function to calculate the Hounsfield Unit and the standard daviation in
    ROI's. This function will be used for the QUASAR phantom only.'''

    # Call the DICOM images
    patient_dicom = load_scans(path)
    # Convert the pixel value
    patient_pixel = get_pixels_hu(patient_dicom)

    # Number of coloums needed in the array
    len_slize = len(patient_pixel[ind_lung])

    # Arrays filled with zeros
    lung     =  np.zeros((5,len_slize))
    dbone    =  np.zeros((5,len_slize))
    water    =  np.zeros((5,len_slize))
    inbone   =  np.zeros((5,len_slize))
    poly     =  np.zeros((5,len_slize))
    cont_q   =  np.zeros((5,len_slize))

    # Only want to use 5 slizes around z = 105 or z = 99 if we are analysing
    # the SECT images. If CT == 2 we have a DECT image, and we will set z = 106,
    # but if not we have a SECT image and we set z = 98
    for i in range(-2, 2+1):
        if CT == 2:
            z = 105 + i
        else:
            z = 98 + i
        image_slize = patient_pixel[:,:,z]

        # Store all the values in an 2d array, find the SD of the whole array
        lung[i   + 2, :] =  image_slize[ind_lung]
        dbone[i  + 2, :] =  image_slize[ind_dbone]
        water[i  + 2, :] =  image_slize[ind_w]
        inbone[i + 2, :] =  image_slize[ind_inbone]
        poly[i   + 2, :] =  image_slize[ind_poly]
        cont_q[i + 2, :] =  image_slize[ind_cont_q]

    # Calculate the mean HU for different material
    mean_lung   = np.mean(lung)
    mean_dbone  = np.mean(dbone)
```

```
    mean_water  = np.mean(water)
    mean_inbone = np.mean(inbone)
    mean_poly   = np.mean(poly)
    mean_cont_q = np.mean(cont_q)

    # Caclculate the standard daviation for the different material
    sd_lung   = np.std(lung, ddof=1)
    sd_dbone  = np.std(dbone, ddof=1)
    sd_water  = np.std(water, ddof=1)
    sd_inbone = np.std(inbone, ddof=1)
    sd_poly   = np.std(poly, ddof=1)
    sd_cont_q = np.std(cont_q, ddof=1)

    return mean_lung, mean_dbone, mean_water, mean_inbone, mean_poly,\
     sd_lung, sd_dbone, sd_water, sd_inbone, sd_poly, mean_cont_q, sd_cont_q

#############################################################################
#####
#####          Get the mean HU and SD for the Gammex phantom          #####
#############################################################################
#####

name_G = ['Blood70', 'Blood40', 'Brain', 'Water', 'Blood', 'Adipose', 'Air']
name_Q = ['Inf. Lung', 'Dense bone', 'Water eq.', 'Inner bone', 'Polyethylene']

# DECT images reconstructed with backprojection
# Set to 'True' if you want to collect the information about the CT images
#reconstructed with backprojection for the Gammex phantom
DECT_images_G = False
if DECT_images_G:
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000C8BE 40keV Gammex//"
    path_2 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000ADE7 50 keV Gammex//"
    path_3 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000971B 60 keV Gammex//"
    path_4 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00001DF5 Gammex C4 serienr 6//"
    path_5 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00007896 80 keV Gammex//"
    path_6 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000116F 90 keV Gammex//"
    path_7 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00001BF3 100 keV Gammex//"
    path_8 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00009BA3 110 keV Gammex//"
    path_9 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000217F 120 keV Gammex//"
    path_10 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000F5A2 130 keV Gammex//"
```

```
    path_11 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000317B 140 keV Gammex//"
    paths = [path_1, path_2, path_3, path_4, path_5, path_6, path_7, \
            path_8, path_9, path_10, path_11]


    # First list is the mean HU and the second in the standard daviation
    # for all the energies (40 keV - 140 keV)
    store_values = [[],[]]
    tesg =np.zeros((11, 7))
    contrast_g = [[],[]]


    # Run through the different paths, and store the data
    for path in paths:
        sd_and_HU = Gammex_DECT_HU_SD(path)
        store_values[0].append(sd_and_HU[0:7])
        store_values[1].append(sd_and_HU[7:14])

        contrast_g[0].append(sd_and_HU[14])
        contrast_g[1].append(sd_and_HU[15])


    # Set the name at a given position
    spreadsheet['A1'] = ('Gammex, DECT, Backprojection')
    spreadsheet['A2'] = ('Mean HU')
    spreadsheet.append(name_G)
    j = 0
    # append the data calculated into the ecxel sheet
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1

    spreadsheet['A16'] = ('Standard daviation')
    spreadsheet.append(name_G)

    j = 0
    for row in tesg:
        spreadsheet.append( store_values[1][j] )
        j += 1



    # Set to 'True' if you want to print the HU for the DECT of the Gammex
    # phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The mean HU for blood70, blood40, brain, water, blood, \
            adipose and air')
        print(store_values[0])
```

```
    # Set to 'True' if you want to print the std for the DECT of the Gammex
    # phantom. If not, keep 'False'
    print_sd = False
    if print_sd:
        print('The std for blood70, blood40, brain, water, blood, \
            adipose and air')
        print(store_values[1])

    print_cont = False
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_g[0])
        print('The STD')
        print(contrast_g[1])


# DECT images reconstructed with iterative method
# Set to 'True' if you want to collect the information about the CT images
#reconstructed with iterative method for the Gammex phantom
DECT_images_AV_G = False
if DECT_images_AV_G:
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00000537 40 keV AV50 Gammex//"
    path_2 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000B825 50 keV AV50 Gammex//"
    path_3 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000098E1 60 keV AV50 Gammex//"
    path_4 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000850F 70 keV AV50 Gammex//"
    path_5 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000EA20 80 keV AV50 Gammex//"
    path_6 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000095F4 90 keV AV50 Gammex//"
    path_7 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00002D91 100 keV AV50 Gammex//"
    path_8 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00001FCA 110 keV AV50 Gammex//"
    path_9 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000024A5 120 keV AV50 Gammex//"
    path_10 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00004869 130 keV AV50 Gammex//"
    path_11 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00005E50 140 keV AV50 Gammex//"
    paths = [path_1, path_2, path_3, path_4, path_5, path_6, path_7, \
    path_8, path_9, path_10, path_11]

    # First list is the mean HU and the second in the standard daviation
    # for all the energies (40 keV - 140 keV)
    store_values = [[],[]]
```

```
contrast_g = [[],[]]

# Run through the different paths, and store the data
for path in paths:
    sd_and_HU = Gammex_DECT_HU_SD(path)
    store_values[0].append(sd_and_HU[0:7])
    store_values[1].append(sd_and_HU[7:14])

    contrast_g[0].append(sd_and_HU[14])
    contrast_g[1].append(sd_and_HU[15])

tesg =np.zeros((11, 7))

spreadsheet['A30'] = ('Gammex, DECT, Iterative reconstruction')
spreadsheet['A31'] = ('Mean HU')
spreadsheet.append(name_G)
j = 0
for row in tesg:
    spreadsheet.append( store_values[0][j] )
    j += 1

#spreadsheet.insert_rows(idx=14, amount=1)
spreadsheet['A45'] = ('Standard daviation')
spreadsheet.append(name_G)

j = 0
for row in tesg:
    spreadsheet.append( store_values[1][j] )
    j += 1

# Set to 'True' if you want to print the HU for the DECT with iterative \
#reconstruction
# of the Gammex phantom. If not, keep 'False'
print_HU = False
if print_HU:
    print('The mean HU for blood70, blood40, brain, water, blood, \
        adipose and air')
    print(store_values[0])

# Set to 'True' if you want to print the std for the DECT with iterative
# reconstruction of the Gammex phantom. If not, keep 'False
print_sd = False
if print_sd:
    print('The std for blood70, blood40, brain, water, blood, adipose and air')
    print(store_values[1])

print_cont = False
if print_cont:
    print('The mean HU of contrast')
    print(contrast_g[0])
```

```
    print('The STD')
    print(contrast_g[1])

#############################################################################
#####
#####          Get the mean HU and SD for the Quasar phantom          #####
#############################################################################
#####

# DECT images reconstructed with backprojections
# Set to 'True' if you want to collect the information about the CT images
# reconstructed with backprojection for the Quasar phantom
QUASAR_DECT_IM = False
if QUASAR_DECT_IM:
   path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000DA64 40 keV Quasar//"
   path_2 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//00000E2E 50keV Quasar//"
   path_3 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000050D 60 keV Quasar//"
   path_4 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00009AE1 70 keV Quasar//"
   path_5 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000025D7 80 keV Quasar//"
   path_6 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000B368 90 keV Quasar//"
   path_7 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00002A4A 100 keV Quasar//"
   path_8 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000E963 110 keV Quasar//"
   path_9 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000CE3F 120keV Quasar//"
   path_10 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00005843 130 keV Quasar//"
   path_11 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000D02A 140 keV Quasar//"
   paths = [path_1, path_2, path_3, path_4, path_5, path_6, path_7, \
   path_8, path_9, path_10, path_11]

   # First list is the mean HU and the second in the standard daviation
   # for all the energies (40 keV - 140 keV)
   store_values = [[],[]]
   contrast_q   = [[],[]]

   # Run through the different paths, and store the data
   for path in paths:
      sd_and_HU = QUASAR_DECT_HU_SD(path, 2)
      store_values[0].append(sd_and_HU[0:5])
      store_values[1].append(sd_and_HU[5:10])
```

```
        contrast_q[0].append(sd_and_HU[10])
        contrast_q[1].append(sd_and_HU[11])

    tesg =np.zeros((11, 5))

    spreadsheet['A59'] = ('Quasar, DECT, Backprojection')
    spreadsheet['A60'] = ('Mean HU')
    spreadsheet.append(name_Q)
    j = 0
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1

    #spreadsheet.insert_rows(idx=14, amount=1)
    spreadsheet['A74'] = ('Standard daviation')
    spreadsheet.append(name_Q)

    j = 0
    for row in tesg:
        spreadsheet.append( store_values[1][j] )
        j += 1

    # Set to 'True' if you want to print the mean HU for the DECT of the Quasar
    # phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The mean HU for inflated lung, dense bone, water , inner bone, \
            and polyethylene')
        print(store_values[0])

    # Set to 'True' if you want to print the std for the DECT of the Quasar
    # phantom. If not, keep 'False'
    print_sd = False
    if print_sd:
        print('The std for lung, dense bone, water, inner bone, and \
            polyethylene')
        print(store_values[1])

    print_cont = False
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_q[0])
        print('The STD')
        print(contrast_q[1])

# DECT images reconstructed with iterative method
# Set to 'True' if you want to collect the information about the CT images
# reconstructed with iterative method for the Quasar phantom
QUASAR_DECT_IM_AV = False
if QUASAR_DECT_IM_AV:
```

```
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000158B 40 keV AV50 Quasar//"
    path_2 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//000045BF 50 keV AV50 Quasar//"
    path_3 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000012BB 60 keV AV50 Quasar//"
    path_4 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000AD20 70 keV AV50 Quasar//"
    path_5 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00007C7C 80 keVAV50 Quasar//"
    path_6 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000991D 90 keV AV50 Quasar//"
    path_7 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00000D16 100 keV AV50 Quasar//"
    path_8 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000012B1 110 keV AV50 Quasar//"
    path_9 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00009128 120keV AV50 Quasar//"
    path_10 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00007803 130 keV AV50 Quasar//"
    path_11 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000B800 140 keV AV50 Quasar//"
    paths = [path_1, path_2, path_3, path_4, path_5, path_6, path_7, \
    path_8, path_9, path_10, path_11]


    # First list is the mean HU and the second in the standard daviation
    # for all the energies (40 keV - 140 keV)
    store_values = [[],[]]
    contrast_q   = [[],[]]

    # Run through the different paths, and store the data
    for path in paths:
        sd_and_HU = QUASAR_DECT_HU_SD(path, 2)
        store_values[0].append(sd_and_HU[0:5])
        store_values[1].append(sd_and_HU[5:10])

        contrast_q[0].append(sd_and_HU[10])
        contrast_q[1].append(sd_and_HU[11])

    tesg =np.zeros((11, 5))

    spreadsheet['A90'] = ('Quasar, DECT, Iterative reconstruction')
    spreadsheet['A91'] = ('Mean HU')
    spreadsheet.append(name_Q)
    j = 0
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1
```

```
    #spreadsheet.insert_rows(idx=14, amount=1)
    spreadsheet['A105'] = ('Standard daviation')
    spreadsheet.append(name_Q)

    j = 0
    for row in tesg:
        spreadsheet.append( store_values[1][j] )
        j += 1



    # Set to 'True' if you want to print the mean HU for the DECT with iterative
    # reconstruction of the Quasar phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The mean HU for inflated lung, dense bone, water , inner bone, \
            and polyethylene')
        print(store_values[0])

    # Set to 'True' if you want to print the std for the DECT with iterative
    # reconstruction of the Quasar phantom. If not, keep 'False'
    print_sd = False
    if print_sd:
        print('The std for lung, dense bone, water, inner bone, and \
            polyethylene')
        print(store_values[1])

    print_cont = False
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_q[0])
        print('The STD')
        print(contrast_q[1])

##############################################################################
#####
#####          Get the mean HU and SD for the SECT images              #####
##############################################################################
#####

# SECT images reconstructed with both methods
# Set to 'True' if you want to collect the information about the CT images
# for the Gammex phantom
SECT_IMAGE_GAMMEX = False
if SECT_IMAGE_GAMMEX:
    # Backprojection
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000C99D Gammex single energi 120kV//"
    # Iterative method
```

```
    path_2 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00000A7D Gammex single energi 120 kV AV50//"

    paths = [path_1, path_2]


    # First list is the mean HU and the second in the standard daviation
    # for both reconstruction algorithms with SECT 120 kV
    store_values = [[],[]]
    contrast_g   = [[],[]]

    # Run through the different paths, and store the data
    for path in paths:
        sd_and_HU = Gammex_DECT_HU_SD(path)
        store_values[0].append(sd_and_HU[0:7])
        store_values[1].append(sd_and_HU[7:14])

        contrast_g[0].append(sd_and_HU[14])
        contrast_g[1].append(sd_and_HU[15])

    tesg =np.zeros((2, 7))

    spreadsheet['A119'] = ('Gammex, SECT, first backprojection then iterative')
    spreadsheet['A120'] = ('Mean HU')
    spreadsheet.append(name_G)
    j = 0
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1

    #spreadsheet.insert_rows(idx=14, amount=1)
    spreadsheet['A125'] = ('Standard daviation')
    spreadsheet.append(name_G)

    j = 0
    for row in tesg:
        spreadsheet.append( store_values[1][j] )
        j += 1



    # Set to 'True' if you want to print the mean HU for the SECT
    # of the Gammex phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The mean HU for blood70, blood40, brain, water, blood, \
            adipose and air')
        print(store_values[0])

    # Set to 'True' if you want to print the standard daviation for the SECT
```

```
    # of the Gammex phantom. If not, keep 'False'
    print_STD = False
    if print_STD:
        print('The std for blood70, blood40, brain, water, blood, adipose and air')
        print(store_values[1])

    print_cont = True
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_g[0])
        print('The STD')
        print(contrast_g[1])

# SECT images reconstructed with both methods
# Set to 'True' if you want to collect the information about the CT images
# for the Quasar phantom
SECT_IMAGE_QUASAR = False
if SECT_IMAGE_QUASAR:
    # bacprojection
    path_1 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000C5F3 Quasar single energi 120kV C10//"
    # iterative method
    path_2 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000ECE8 Quasar single energi 120 kV AV50//"

    paths = [path_1, path_2]

    # First list is the mean HU and the second in the standard daviation
    # for both reconstruction algorithms with SECT 120 kV
    store_values = [[],[]]
    contrast_q   = [[],[]]

    # Run through the different paths, and store the data
    for path in paths:
        sd_and_HU = QUASAR_DECT_HU_SD(path, 0)
        store_values[0].append(sd_and_HU[0:5])
        store_values[1].append(sd_and_HU[5:10])

        contrast_q[0].append(sd_and_HU[10])
        contrast_q[1].append(sd_and_HU[11])

    tesg =np.zeros((2, 5))

    spreadsheet['A130'] = ('Quasar, SECT, First bacprojection then iterative.')
    spreadsheet['A131'] = ('Mean HU')
    spreadsheet.append(name_Q)
    j = 0
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1
```

```
#spreadsheet.insert_rows(idx=14, amount=1)
spreadsheet['A136'] = ('Standard daviation')
spreadsheet.append(name_Q)

j = 0
for row in tesg:
    spreadsheet.append( store_values[1][j] )
    j += 1

# Save the data in a file in the same folder as the program is located
book.save('data_from_DICOM.xlsx')

# Set to 'True' if you want to print the mean HU for the SECT
# of the Quasar phantom. If not, keep 'False'
print_HU = False
if print_HU:
    print('The mean HU for inflated lung, dense bone, water , inner bone,\
     and polyethylene')
    print(store_values[0])

# Set to 'True' if you want to print the standard daviation for the SECT
# of the Quasar phantom. If not, keep 'False'
print_HU = False
if print_HU:
    print('The std for inflated lung, dense bone, water , inner bone, \
        and polyethylene')
    print(store_values[1])

print_cont = True
if print_cont:
    print('The mean HU of contrast')
    print(contrast_q[0])
    print('The STD')
    print(contrast_q[1])
```

Python code creating a tablet for extracting HU values from the anthropomorphic phantom. Every calculation is set to False as default.

```
import pydicom as dcm
import numpy as np
import matplotlib.pyplot as plt
import glob
from openpyxl import Workbook
#import scipy as sp
#import scipy.interpolate

# a function to read the DICOM files, and sort them in correct order
def load_scans(path):
        files = glob.glob(path+"*") #Søker etter CT-filer i katalog. Evt bruk "*"

        zdim = len(files) # Antall filer = antall snitt

        test = dcm.dcmread(files[0]).pixel_array #leser ut et vilkårlig bilde, xog
        xdim = len(test[0]) # antall x = antall pixler i x
        ydim = len(test[1]) # antall y = antall pixler i y
        #print(test.shape)

        images = np.empty([xdim, ydim, zdim])
        sli_loc = np.empty([zdim])
        for z in range(zdim):
            sli_loc[z] = dcm.dcmread(files[z]).SliceLocation
        # Leste inn slice location, ikke riktig rekkefølge

        index_sort = np.argsort(sli_loc)

        for z in range(zdim):
            index = index_sort[z] # Leser inn bildene sortert etter slice location
            image = dcm.dcmread(files[index]).pixel_array
            images[:, :, z]= image

            #images = images.astype(np.int16)

        #print(images)
        return images

# The DICOM images have an Rescale intercept of -1024, and a Rescale Slope of 1,
# so to find the Hounsfield Unit in the DICOM image we have to use:
# U = m * SV + b
# where U = HU, m = Rescale Slope, SV = given value in the voxel, and
# b = Rescale intercept
def get_pixels_hu(scans):
    ima = scans


    ima += np.int16(-1024)
```

```
    return np.array(ima, dtype=np.int16)

# Show every 7th DICOM file, to figure out which slize we want to use
def sample_stack(stack, rows=4, cols=4, start_with=1, show_every=7):
    fig,ax = plt.subplots(rows,cols,figsize=[10,10])
    for i in range(rows*cols):
        ind = start_with + i*show_every
        ax[int(i/rows),int(i % rows)].set_title('slice %d' % ind)
        ax[int(i/rows),int(i % rows)].imshow(stack[:,:, ind], cmap=plt.cm.bone)
        ax[int(i/rows),int(i % rows)].axis('off')
    plt.show()

# A template for the circular ROI's in the GAMMEX phantom, returns the indexes
# where the ciruclar ROI's are located. This template is for the GAMMEX phantom
def template_gammex():

    # number of pixels in the radius
    ra = 10

    # choose which slize we want to use for the template
    dummy = patient_pixel[:,:, 92]

    '''
    circ = np.empty([512, 512])

    x0 = 435
    y0 = 252

    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                circ[i  + y0, j  + x0] = 12

    ind_cir = np.where( circ == 12)
    #print(ind_cir)

    #for i in range()

    dummy = patient_dicom[:,:, 92]

    dummy[ind_cir] = 0

    plt.figure()
    plt.imshow(dummy)
    plt.show()

    print(np.mean(dummy[ind_cir]))
```

```
print(np.std(dummy[ind_cir]))
'''
#######################################################################

# find the index for the first circle
x1 = 133 #122
y1 = 131 #122

# make a two dimentional array with size 512x512 filled with zeros
circle1 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 1. (array)
            circle1[i + y1, j + x1] = 1

# collect the indexes where the array is 1 (our wanted circle)
ind_circle1 = np.where( circle1 == 1)

#######################################################################

# find the index for the secound circle
x2 = 380 #372
y2 = 129 #118

# make a two dimentional array with size 512x512 filled with zeros
circle2 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 2.  (array)
            circle2[i + y2, j + x2] = 2
# collect the indexes where the array is 1 (our wanted circle)
ind_circle2 = np.where( circle2 == 2)

#######################################################################

# find the index for the third circle
x3 = 189 #179
y3 = 190 #179

# make a two dimentional array with size 512x512 filled with zeros
```

```
circle3 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 3. (array)
            circle3[i + y3, j + x3] = 3
# collect the indexes where the array is 1 (our wanted circle)
ind_circle3 = np.where( circle3 == 3)

#########################################################################

# find the index for the fourth circle
x4 = 255
y4 = 160

# make a two dimentional array with size 512x512 filled with zeros
circle4 = np.zeros([512,512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 4 (array)
            circle4[i + y4, j + x4] = 4

ind_circle4 = np.where( circle4 == 4)

#########################################################################

# find the index for the fifth circle
x5 = 83 #74
y5 = 256 #246

# make a two dimentional array with size 512x512 filled with zeros
circle5 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 5 (array)
            circle5[i + y5, j + x5] = 5
```

```
# collect the indexes where the array is 1 (our wanted circle)
ind_circle5 = np.where( circle5 == 5)

#########################################################################

# find the index for the sixth circle
x6 = 353 #343
y6 = 250 #240

# make a two dimentional array with size 512x512 filled with zeros
circle6 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 6 (array)
            circle6[i + y6, j + x6] = 6
# collect the indexes where the array is 1 (our wanted circle)
ind_circle6 = np.where( circle6 == 6)

#########################################################################

# find the index for the seventh circle
x7 = 194 #184
y7 = 321 #312

# make a two dimentional array with size 512x512 filled with zeros
circle7 = np.zeros([512, 512])

# creat mask in a circular pathern around the centrum of a choosen index
for i in range(-ra, ra+1):
    for j in range(-ra, ra+1):
        r = np.sqrt( (i**2.0 ) + (j**2.0) )

        if r <= ra:
            #fill the indexes that is a part of the circle with 7. (array)
            circle7[i + y7, j + x7] = 7
# collect the indexes where the array is 1 (our wanted circle)
ind_circle7 = np.where( circle7 == 7)

#########################################################################

# find the index for the eight circle
x8 = 247 #184
y8 = 47 #312

# make a two dimentional array with size 512x512 filled with zeros
```

```
    circle8 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 8. (array)
                circle8[i + y8, j + x8] = 8
    # collect the indexes where the array is 1 (our wanted circle)
    ind_circle8 = np.where( circle8 == 8)


    # test to check that the circle is in the right spot.
    # Set to 'True' if you want to plot the image
    test_spot = False
    if test_spot:
        dummy[ind_circle8] = 1200

        plt.figure()
        plt.imshow(dummy, cmap=plt.cm.bone)
        plt.show()

        #print(np.mean(dummy[ind_circle1])) # can only be used if dummy = -1200
        #print(np.std(dummy[ind_circle1])) # can only be used if dummy = -1200

    return ind_circle1, ind_circle2, ind_circle3, ind_circle4, ind_circle5,\
     ind_circle6, ind_circle7, ind_circle8

# A template for the circular ROI's in the QUASAR phantom, returns the indexes
# where the ciruclar ROI's are located. This template is for the GAMMEX phantom
def template_quasar():
    # number of pixels in the radius (same for both phantoms)
    ra = 10

    # choose what slize we want to use as the template
    dummy = patient_pixel_q[:,:, 106]

    '''
    circ = np.empty([512, 512])

    x0 = 257
    y0 = 252

    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
```

```
            circ[i  + y0, j  + x0] = 12

    ind_cir = np.where( circ == 12)
    #print(ind_cir)
    '''


###############################################################################
#

    # find the index for the first circle
    x1 = 257
    y1 = 204

    # make a two dimentional array with size 512x512 filled with zeros
    circle1 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 1. (array)
                circle1[i + y1, j + x1] = 1
    # collect the indexes where the array is 1 (our wanted circle)
    ind_circle1 = np.where( circle1 == 1)


###############################################################################
#
    # find the index for the secound circle
    x2 = 305
    y2 = 252

    # make a two dimentional array with size 512x512 filled with zeros
    circle2 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 2. (array)
                circle2[i + y2, j + x2] = 2
    # collect the indexes where the array is 2 (our wanted circle)
    ind_circle2 = np.where( circle2 == 2)
```

```
#############################################################################
#
   # find the index for the third circle
   x3 = 257
   y3 = 300

   # make a two dimentional array with size 512x512 filled with zeros
   circle3 = np.zeros([512, 512])

   # creat mask in a circular pathern around the centrum of a choosen index
   for i in range(-ra, ra+1):
      for j in range(-ra, ra+1):
         r = np.sqrt( (i**2.0 ) + (j**2.0) )

         if r <= ra:
            #fill the indexes that is a part of the circle with 3. (array)
            circle3[i + y3, j + x3] = 3
   # collect the indexes where the array is 3 (our wanted circle)
   ind_circle3 = np.where( circle3 == 3)


#############################################################################
#
   # find the index for the fourth circle
   x4 = 210
   y4 = 252

   # make a two dimentional array with size 512x512 filled with zeros
   circle4 = np.zeros([512,512])

   # creat mask in a circular pathern around the centrum of a choosen index
   for i in range(-ra, ra+1):
      for j in range(-ra, ra+1):
         r = np.sqrt( (i**2.0 ) + (j**2.0) )

         if r <= ra:
            #fill the indexes that is a part of the circle with 4. (array)
            circle4[i + y4, j + x4] = 4

   # collect the indexes where the array is 4 (our wanted circle)
   ind_circle4 = np.where( circle4 == 4)



#############################################################################
#
   # find the index for the fifth circle
   x5 = 257
   y5 = 252
```

```
    # make a two dimentional array with size 512x512 filled with zeros
    circle5 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 5. (array)
                circle5[i + y5, j + x5] = 5

    # collect the indexes where the array is 5 (our wanted circle)
    ind_circle5 = np.where( circle5 == 5)


##############################################################################
#

    # find the index for the sixth circle
    x6 = 257
    y6 = 160

    # make a two dimentional array with size 512x512 filled with zeros
    circle6 = np.zeros([512, 512])

    # creat mask in a circular pathern around the centrum of a choosen index
    for i in range(-ra, ra+1):
        for j in range(-ra, ra+1):
            r = np.sqrt( (i**2.0 ) + (j**2.0) )

            if r <= ra:
                #fill the indexes that is a part of the circle with 1. (array)
                circle6[i + y6, j + x6] = 6
    # collect the indexes where the array is 1 (our wanted circle)
    ind_circle6 = np.where( circle6 == 6)

    # test to check that the circle is in the right spot.
    # Set to 'True' if you want to plot the image
    check_temp = True
    if check_temp:
        dummy = patient_pixel_q[:,:, 106]

        dummy[ind_circle6] = 1200

        plt.figure(1)
        plt.imshow(dummy)
        plt.show()
```

```
    #print(np.mean(dummy[ind_cir]))
    #print(np.std(dummy[ind_cir]))

    return ind_circle1, ind_circle2, ind_circle3, ind_circle4, ind_circle5,\
     ind_circle6

# Able to save the data to a excel file
book = Workbook()

# Activate a sheet in the Workbook, able to write into it
spreadsheet = book.active

# Load the scans for the template
#############################################################################
#####
#####              G A M M E X   P H A N T O M                  #####
#############################################################################
#####

# path1 is a path to a Gammex phantom
path1="//Users/Sandra/Documents/UIO/Master/Masteroppgave/Method/CT-GE/\
PixPadZero/00000A7D Gammex single energi 120 kV AV50//"
# Katalog der CT-bildene ligger (NB ingen andre filer bør ligge der)

# start with the gammex phantom, loading the DICOM images from the map
patient_dicom = load_scans(path1)
# correctiong the pixel value to HU
patient_pixel = get_pixels_hu(patient_dicom)

# Check the stack, to find an image we can center around
imgs_to_process = patient_pixel

check_stack = False
if check_stack:
    sample_stack(imgs_to_process)

# Find the index for the ROI location for the different materials.
# A template to use on the DICOM images.
# Set as global variables
ind_blood70, ind_blood40, ind_brain, ind_water, ind_blood, ind_adipose, \
ind_air, ind_cont_g = template_gammex()

# From the stack image we know that the gammex has nice images around slize 92.
# Will use 5 slizes around number 92

#############################################################################
#####
#####              Q U A S A R   P H A N T O M                  #####
#############################################################################
#####
```

```
#path2 is a path to a quasar phantom
Path2 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000AD20 70 keV AV50 Quasar//"
#Path3 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//\
#CT-GE//PixPadZero//0000ECE8 Quasar single energi 120 kV AV50//"

# Start by loading the DICOM images from path2
patient_dicom_q = load_scans(Path2)
patient_pixel_q = get_pixels_hu(patient_dicom_q)

# Check the stack, to find an image we can center around
check_stack_q = False
if check_stack_q:
    sample_stack(patient_pixel_q)

# From stack check the interesting inserts looks best around slize 106.
# Will use 5 slizes around number 106
# Can alsow see that for the SECT we have to choose around slize 99 not 106
# But the location of the ROI's is still the same

# Find the index for the ROI location for the different materials.
# A template to use on the DICOM images.
# set as global variables
ind_lung, ind_dbone, ind_w, ind_inbone, ind_poly, ind_cont_q = template_quasar()

#############################################################################
#####
#############################################################################
#####
#####                                                 #####
#####          FIND THE MEAN HU AND STD IN THE ROIS          #####
#####                                                 #####
#############################################################################
#####
#############################################################################
#####

def Gammex_DECT_HU_SD(path):
    '''A function to calculate the Hounsfield Unit and the standard daviation in
    ROI's. This function will be used for the Gammex phantom only.'''

    # Call the DICOM images
    patient_dicom = load_scans(path)
    # Correct the pixel value
    patient_pixel = get_pixels_hu(patient_dicom)

    # Number of coloums needed in the array
    len_slize = len(patient_pixel[ind_blood70])
```

```
    # Arrays filled with zeros
    blood70     =  np.zeros((5,len_slize))
    blood40     =  np.zeros((5,len_slize))
    brain       =  np.zeros((5,len_slize))
    water       =  np.zeros((5,len_slize))
    blood       =  np.zeros((5,len_slize))
    adipose     =  np.zeros((5,len_slize))
    air         =  np.zeros((5,len_slize))
    contr_g     =  np.zeros((5,len_slize))

    # Only want to use 5 slizes around z = 92
    for i in range(-2, 2+1):
        z = 92 + i
        image_slize = patient_pixel[:,:,z]

        # Store all the values in an 2d array, find the SD of the whole array
        blood70[i + 2, :]    =  image_slize[ind_blood70]
        blood40[i + 2, :]    =  image_slize[ind_blood40]
        brain[i   + 2, :]    =  image_slize[ind_brain]
        water[i   + 2, :]    =  image_slize[ind_water]
        blood[i   + 2, :]    =  image_slize[ind_blood]
        adipose[i + 2, :]    =  image_slize[ind_adipose]
        air[i     + 2, :]    =  image_slize[ind_air]
        contr_g[i   + 2, :]  =  image_slize[ind_cont_g]

    # Calculate the mean HU for different material
    mean_blood70 = np.mean(blood70)
    mean_blood40 = np.mean(blood40)
    mean_brain   = np.mean(brain)
    mean_blood   = np.mean(blood)
    mean_water   = np.mean(water)
    mean_adipose = np.mean(adipose)
    mean_air     = np.mean(air)
    mean_cont_g  = np.mean(contr_g)

    # Caclculate the standard daviation for the different material
    sd_blood70  = np.std(blood70, ddof=1)
    sd_blood40  = np.std(blood40, ddof=1)
    sd_brain    = np.std(brain, ddof=1)
    sd_blood    = np.std(blood, ddof=1)
    sd_water    = np.std(water, ddof=1)
    sd_adipose  = np.std(adipose, ddof=1)
    sd_air      = np.std(air, ddof=1)
    sd_cont_g   = np.std(contr_g, ddof=1)

    return mean_blood70, mean_blood40, mean_brain, mean_water, mean_blood, \
    mean_adipose, mean_air, sd_blood70, sd_blood40, sd_brain, sd_blood, sd_water,
sd_adipose, \
     sd_air, mean_cont_g, sd_cont_g
```

```
def QUASAR_DECT_HU_SD(path, CT):
    '''A function to calculate the Hounsfield Unit and the standard daviation in
    ROI's. This function will be used for the QUASAR phantom only.'''

    # Call the DICOM images
    patient_dicom = load_scans(path)
    # Convert the pixel value
    patient_pixel = get_pixels_hu(patient_dicom)

    # Number of coloums needed in the array
    len_slize = len(patient_pixel[ind_lung])

    # Arrays filled with zeros
    lung    =  np.zeros((5,len_slize))
    dbone   =  np.zeros((5,len_slize))
    water   =  np.zeros((5,len_slize))
    inbone  =  np.zeros((5,len_slize))
    poly    =  np.zeros((5,len_slize))
    cont_q  =  np.zeros((5,len_slize))

    # Only want to use 5 slizes around z = 105 or z = 99 if we are analysing
    # the SECT images. If CT == 2 we have a DECT image, and we will set z = 106,
    # but if not we have a SECT image and we set z = 98
    for i in range(-2, 2+1):
        if CT == 2:
            z = 105 + i
        else:
            z = 98 + i
        image_slize = patient_pixel[:,:,z]

        # Store all the values in an 2d array, find the SD of the whole array
        lung[i  + 2, :] =  image_slize[ind_lung]
        dbone[i  + 2, :] =  image_slize[ind_dbone]
        water[i  + 2, :] =  image_slize[ind_w]
        inbone[i + 2, :] =  image_slize[ind_inbone]
        poly[i  + 2, :] =  image_slize[ind_poly]
        cont_q[i + 2, :] =  image_slize[ind_cont_q]

    # Calculate the mean HU for different material
    mean_lung   = np.mean(lung)
    mean_dbone  = np.mean(dbone)
    mean_water  = np.mean(water)
    mean_inbone = np.mean(inbone)
    mean_poly   = np.mean(poly)
    mean_cont_q = np.mean(cont_q)

    # Caclculate the standard daviation for the different material
    sd_lung   = np.std(lung, ddof=1)
    sd_dbone  = np.std(dbone, ddof=1)
```

```
    sd_water  = np.std(water, ddof=1)
    sd_inbone = np.std(inbone, ddof=1)
    sd_poly   = np.std(poly, ddof=1)
    sd_cont_q = np.std(cont_q, ddof=1)

    return mean_lung, mean_dbone, mean_water, mean_inbone, mean_poly,\
     sd_lung, sd_dbone, sd_water, sd_inbone, sd_poly, mean_cont_q, sd_cont_q

##########################################################################
#####
#####          Get the mean HU and SD for the Gammex phantom          #####
##########################################################################
#####

name_G = ['Blood70', 'Blood40', 'Brain', 'Water', 'Blood', 'Adipose', 'Air']
name_Q = ['Inf. Lung', 'Dense bone', 'Water eq.', 'Inner bone', 'Polyethylene']

# DECT images reconstructed with backprojection
# Set to 'True' if you want to collect the information about the CT images
#reconstructed with backprojection for the Gammex phantom
DECT_images_G = False
if DECT_images_G:
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000C8BE 40keV Gammex//"
    path_2 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000ADE7 50 keV Gammex//"
    path_3 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000971B 60 keV Gammex//"
    path_4 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00001DF5 Gammex C4 serienr 6//"
    path_5 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00007896 80 keV Gammex//"
    path_6 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000116F 90 keV Gammex//"
    path_7 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00001BF3 100 keV Gammex//"
    path_8 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00009BA3 110 keV Gammex//"
    path_9 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000217F 120 keV Gammex//"
    path_10 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000F5A2 130 keV Gammex//"
    path_11 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000317B 140 keV Gammex//"
    paths = [path_1, path_2, path_3, path_4, path_5, path_6, path_7, \
            path_8, path_9, path_10, path_11]


    # First list is the mean HU and the second in the standard daviation
    # for all the energies (40 keV - 140 keV)
```

```
store_values = [[],[]]
tesg =np.zeros((11, 7))
contrast_g = [[],[]]


# Run through the different paths, and store the data
for path in paths:
    sd_and_HU = Gammex_DECT_HU_SD(path)
    store_values[0].append(sd_and_HU[0:7])
    store_values[1].append(sd_and_HU[7:14])

    contrast_g[0].append(sd_and_HU[14])
    contrast_g[1].append(sd_and_HU[15])


# Set the name at a given position
spreadsheet['A1'] = ('Gammex, DECT, Backprojection')
spreadsheet['A2'] = ('Mean HU')
spreadsheet.append(name_G)
j = 0
# append the data calculated into the ecxel sheet
for row in tesg:
    spreadsheet.append( store_values[0][j] )
    j += 1

spreadsheet['A16'] = ('Standard daviation')
spreadsheet.append(name_G)

j = 0
for row in tesg:
    spreadsheet.append( store_values[1][j] )
    j += 1



# Set to 'True' if you want to print the HU for the DECT of the Gammex
# phantom. If not, keep 'False'
print_HU = False
if print_HU:
    print('The mean HU for blood70, blood40, brain, water, blood, \
        adipose and air')
    print(store_values[0])

# Set to 'True' if you want to print the std for the DECT of the Gammex
# phantom. If not, keep 'False'
print_sd = False
if print_sd:
    print('The std for blood70, blood40, brain, water, blood, \
        adipose and air')
    print(store_values[1])
```

```
    print_cont = False
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_g[0])
        print('The STD')
        print(contrast_g[1])


# DECT images reconstructed with iterative method
# Set to 'True' if you want to collect the information about the CT images
#reconstructed with iterative method for the Gammex phantom
DECT_images_AV_G = False
if DECT_images_AV_G:
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00000537 40 keV AV50 Gammex//"
    path_2 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000B825 50 keV AV50 Gammex//"
    path_3 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000098E1 60 keV AV50 Gammex//"
    path_4 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000850F 70 keV AV50 Gammex//"
    path_5 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000EA20 80 keV AV50 Gammex//"
    path_6 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000095F4 90 keV AV50 Gammex//"
    path_7 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00002D91 100 keV AV50 Gammex//"
    path_8 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00001FCA 110 keV AV50 Gammex//"
    path_9 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000024A5 120 keV AV50 Gammex//"
    path_10 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00004869 130 keV AV50 Gammex//"
    path_11 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00005E50 140 keV AV50 Gammex//"
    paths = [path_1, path_2, path_3, path_4, path_5, path_6, path_7, \
    path_8, path_9, path_10, path_11]

    # First list is the mean HU and the second in the standard daviation
    # for all the energies (40 keV - 140 keV)
    store_values = [[],[]]
    contrast_g = [[],[]]

    # Run through the different paths, and store the data
    for path in paths:
        sd_and_HU = Gammex_DECT_HU_SD(path)
        store_values[0].append(sd_and_HU[0:7])
        store_values[1].append(sd_and_HU[7:14])
```

```
    contrast_g[0].append(sd_and_HU[14])
    contrast_g[1].append(sd_and_HU[15])

tesg =np.zeros((11, 7))

spreadsheet['A30'] = ('Gammex, DECT, Iterative reconstruction')
spreadsheet['A31'] = ('Mean HU')
spreadsheet.append(name_G)
j = 0
for row in tesg:
    spreadsheet.append( store_values[0][j] )
    j += 1

#spreadsheet.insert_rows(idx=14, amount=1)
spreadsheet['A45'] = ('Standard daviation')
spreadsheet.append(name_G)

j = 0
for row in tesg:
    spreadsheet.append( store_values[1][j] )
    j += 1

# Set to 'True' if you want to print the HU for the DECT with iterative \
#reconstruction
# of the Gammex phantom. If not, keep 'False'
print_HU = False
if print_HU:
    print('The mean HU for blood70, blood40, brain, water, blood, \
       adipose and air')
    print(store_values[0])

# Set to 'True' if you want to print the std for the DECT with iterative
# reconstruction of the Gammex phantom. If not, keep 'False
print_sd = False
if print_sd:
    print('The std for blood70, blood40, brain, water, blood, adipose and air')
    print(store_values[1])

print_cont = False
if print_cont:
    print('The mean HU of contrast')
    print(contrast_g[0])
    print('The STD')
    print(contrast_g[1])

#############################################################################
#####
#####          Get the mean HU and SD for the Quasar phantom          #####
#############################################################################
#####
```

```
# DECT images reconstructed with backprojections
# Set to 'True' if you want to collect the information about the CT images
# reconstructed with backprojection for the Quasar phantom
QUASAR_DECT_IM = False
if QUASAR_DECT_IM:
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000DA64 40 keV Quasar//"
    path_2 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//00000E2E 50keV Quasar//"
    path_3 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000050D 60 keV Quasar//"
    path_4 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00009AE1 70 keV Quasar//"
    path_5 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000025D7 80 keV Quasar//"
    path_6 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000B368 90 keV Quasar//"
    path_7 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00002A4A 100 keV Quasar//"
    path_8 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000E963 110 keV Quasar//"
    path_9 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000CE3F 120keV Quasar//"
    path_10 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00005843 130 keV Quasar//"
    path_11 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000D02A 140 keV Quasar//"
    paths = [path_1, path_2, path_3, path_4, path_5, path_6, path_7, \
    path_8, path_9, path_10, path_11]

    # First list is the mean HU and the second in the standard daviation
    # for all the energies (40 keV - 140 keV)
    store_values = [[],[]]
    contrast_q   = [[],[]]

    # Run through the different paths, and store the data
    for path in paths:
        sd_and_HU = QUASAR_DECT_HU_SD(path, 2)
        store_values[0].append(sd_and_HU[0:5])
        store_values[1].append(sd_and_HU[5:10])

        contrast_q[0].append(sd_and_HU[10])
        contrast_q[1].append(sd_and_HU[11])

    tesg =np.zeros((11, 5))

    spreadsheet['A59'] = ('Quasar, DECT, Backprojection')
    spreadsheet['A60'] = ('Mean HU')
    spreadsheet.append(name_Q)
```

```
    j = 0
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1

    #spreadsheet.insert_rows(idx=14, amount=1)
    spreadsheet['A74'] = ('Standard daviation')
    spreadsheet.append(name_Q)

    j = 0
    for row in tesg:
        spreadsheet.append( store_values[1][j] )
        j += 1

    # Set to 'True' if you want to print the mean HU for the DECT of the Quasar
    # phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The mean HU for inflated lung, dense bone, water , inner bone, \
            and polyethylene')
        print(store_values[0])

    # Set to 'True' if you want to print the std for the DECT of the Quasar
    # phantom. If not, keep 'False'
    print_sd = False
    if print_sd:
        print('The std for lung, dense bone, water, inner bone, and \
            polyethylene')
        print(store_values[1])

    print_cont = False
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_q[0])
        print('The STD')
        print(contrast_q[1])

# DECT images reconstructed with iterative method
# Set to 'True' if you want to collect the information about the CT images
# reconstructed with iterative method for the Quasar phantom
QUASAR_DECT_IM_AV = False
if QUASAR_DECT_IM_AV:
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000158B 40 keV AV50 Quasar//"
    path_2 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//000045BF 50 keV AV50 Quasar//"
    path_3 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000012BB 60 keV AV50 Quasar//"
    path_4 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000AD20 70 keV AV50 Quasar//"
```

```
    path_5 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00007C7C 80 keVAV50 Quasar//"
    path_6 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000991D 90 keV AV50 Quasar//"
    path_7 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00000D16 100 keV AV50 Quasar//"
    path_8 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//000012B1 110 keV AV50 Quasar//"
    path_9 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00009128 120keV AV50 Quasar//"
    path_10 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00007803 130 keV AV50 Quasar//"
    path_11 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000B800 140 keV AV50 Quasar//"
    paths = [path_1, path_2, path_3, path_4, path_5, path_6, path_7, \
    path_8, path_9, path_10, path_11]


    # First list is the mean HU and the second in the standard daviation
    # for all the energies (40 keV - 140 keV)
    store_values = [[],[]]
    contrast_q   = [[],[]]

    # Run through the different paths, and store the data
    for path in paths:
        sd_and_HU = QUASAR_DECT_HU_SD(path, 2)
        store_values[0].append(sd_and_HU[0:5])
        store_values[1].append(sd_and_HU[5:10])

        contrast_q[0].append(sd_and_HU[10])
        contrast_q[1].append(sd_and_HU[11])

    tesg =np.zeros((11, 5))

    spreadsheet['A90'] = ('Quasar, DECT, Iterative reconstruction')
    spreadsheet['A91'] = ('Mean HU')
    spreadsheet.append(name_Q)
    j = 0
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1

    #spreadsheet.insert_rows(idx=14, amount=1)
    spreadsheet['A105'] = ('Standard daviation')
    spreadsheet.append(name_Q)

    j = 0
    for row in tesg:
        spreadsheet.append( store_values[1][j] )
        j += 1
```

```
    # Set to 'True' if you want to print the mean HU for the DECT with iterative
    # reconstruction of the Quasar phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The mean HU for inflated lung, dense bone, water , inner bone, \
            and polyethylene')
        print(store_values[0])

    # Set to 'True' if you want to print the std for the DECT with iterative
    # reconstruction of the Quasar phantom. If not, keep 'False'
    print_sd = False
    if print_sd:
        print('The std for lung, dense bone, water, inner bone, and \
            polyethylene')
        print(store_values[1])

    print_cont = False
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_q[0])
        print('The STD')
        print(contrast_q[1])

################################################################################
#####
#####          Get the mean HU and SD for the SECT images              #####
################################################################################
#####

# SECT images reconstructed with both methods
# Set to 'True' if you want to collect the information about the CT images
# for the Gammex phantom
SECT_IMAGE_GAMMEX = False
if SECT_IMAGE_GAMMEX:
    # Backprojection
    path_1 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000C99D Gammex single energi 120kV//"
    # Iterative method
    path_2 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//00000A7D Gammex single energi 120 kV AV50//"

    paths = [path_1, path_2]

    # First list is the mean HU and the second in the standard daviation
    # for both reconstruction algorithms with SECT 120 kV
    store_values = [[],[]]
    contrast_g   = [[],[]]

    # Run through the different paths, and store the data
```

```
    for path in paths:
        sd_and_HU = Gammex_DECT_HU_SD(path)
        store_values[0].append(sd_and_HU[0:7])
        store_values[1].append(sd_and_HU[7:14])

        contrast_g[0].append(sd_and_HU[14])
        contrast_g[1].append(sd_and_HU[15])

    tesg =np.zeros((2, 7))

    spreadsheet['A119'] = ('Gammex, SECT, first backprojection then iterative')
    spreadsheet['A120'] = ('Mean HU')
    spreadsheet.append(name_G)
    j = 0
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1

    #spreadsheet.insert_rows(idx=14, amount=1)
    spreadsheet['A125'] = ('Standard daviation')
    spreadsheet.append(name_G)

    j = 0
    for row in tesg:
        spreadsheet.append( store_values[1][j] )
        j += 1

    # Set to 'True' if you want to print the mean HU for the SECT
    # of the Gammex phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The mean HU for blood70, blood40, brain, water, blood, \
            adipose and air')
        print(store_values[0])

    # Set to 'True' if you want to print the standard daviation for the SECT
    # of the Gammex phantom. If not, keep 'False'
    print_STD = False
    if print_STD:
        print('The std for blood70, blood40, brain, water, blood, adipose and air')
        print(store_values[1])

    print_cont = True
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_g[0])
        print('The STD')
        print(contrast_g[1])

# SECT images reconstructed with both methods
```

```
# Set to 'True' if you want to collect the information about the CT images
# for the Quasar phantom
SECT_IMAGE_QUASAR = False
if SECT_IMAGE_QUASAR:
    # bacprojection
    path_1 = "//Users//Sandra//Documents//UIO/Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000C5F3 Quasar single energi 120kV C10//"
    # iterative method
    path_2 = "//Users//Sandra//Documents//UIO//Master//Masteroppgave//Method//CT-
GE//PixPadZero//0000ECE8 Quasar single energi 120 kV AV50//"

    paths = [path_1, path_2]

    # First list is the mean HU and the second in the standard daviation
    # for both reconstruction algorithms with SECT 120 kV
    store_values = [[],[]]
    contrast_q   = [[],[]]

    # Run through the different paths, and store the data
    for path in paths:
        sd_and_HU = QUASAR_DECT_HU_SD(path, 0)
        store_values[0].append(sd_and_HU[0:5])
        store_values[1].append(sd_and_HU[5:10])

        contrast_q[0].append(sd_and_HU[10])
        contrast_q[1].append(sd_and_HU[11])

    tesg =np.zeros((2, 5))

    spreadsheet['A130'] = ('Quasar, SECT, First bacprojection then iterative.')
    spreadsheet['A131'] = ('Mean HU')
    spreadsheet.append(name_Q)
    j = 0
    for row in tesg:
        spreadsheet.append( store_values[0][j] )
        j += 1

    #spreadsheet.insert_rows(idx=14, amount=1)
    spreadsheet['A136'] = ('Standard daviation')
    spreadsheet.append(name_Q)

    j = 0
    for row in tesg:
        spreadsheet.append( store_values[1][j] )
        j += 1

    # Save the data in a file in the same folder as the program is located
    book.save('data_from_DICOM.xlsx')

    # Set to 'True' if you want to print the mean HU for the SECT
```

```
    # of the Quasar phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The mean HU for inflated lung, dense bone, water , inner bone,\
         and polyethylene')
        print(store_values[0])

    # Set to 'True' if you want to print the standard daviation for the SECT
    # of the Quasar phantom. If not, keep 'False'
    print_HU = False
    if print_HU:
        print('The std for inflated lung, dense bone, water , inner bone, \
            and polyethylene')
        print(store_values[1])

    print_cont = True
    if print_cont:
        print('The mean HU of contrast')
        print(contrast_q[0])
        print('The STD')
        print(contrast_q[1])
```