2D dosimetry and radiobiological modelling in GRID therapy

Bjørg Vårli Håland

Biological and Medical Physics
Department of Physics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

June 2020

Acknowledgements

The work presented in this thesis was carried out at the section for Biophysics and Medical physics at the Department of Physics, University of Oslo.

First, I would like to sincerely thank my supervisors Eirik Malinen, Nina F. Edin and Delmon Arous for their guidance and support throughout this work. A special thanks goes to Delmon Arous for all assistance and motivation, and for always being ready to spend time answering my questions. I am truly grateful for every discussion and all help, without it none of this would have been possible.

I would also like to thank Magnus Børsting for good and helpful cooperation during the work on this project, and lastly, a thanks goes to all the students and employees at the Biophysics and Medical physics section, for the good, relaxed and supportive atmosphere. I have really enjoyed my time there.

June 2020

Bjørg Vårli Håland

Summary

The goal of radiotherapy is to eradicate tumors while causing as little damage and side effects as possible. Conventional radiotherapy aims to give a homogenous dose to a target (the tumor), but the radiation fields necessary to achieve this will inevitably also give dose to healthy normal tissue, as they cover vital and sensitive organs. Suggestions have been made that by irradiating with a spatially fractionated pattern, i.e. giving high very high doses in a spot-like pattern with low or no doses in between, we can achieve the same amount of tumor killing while sparing more of the surrounding tissue. This modality is called SFRT – spatially fractionated radiation therapy, or simply GRID therapy, as the dose pattern is in form of a grid. Although some promising results with GRID therapy have been showed, the mechanisms behind it are not fully understood.

This thesis presents a methodology for obtaining the dose distribution in cell culture flasks when using GRID irradiation, both with X-rays and protons. The GRID configuration was in form of a striped field, using a tungsten collimator with stripes of 10 mm shielding and 5 mm openings in between. The goal was to establish the dose distribution in cell culture flasks when irradiated using this GRID collimator, and further use this dose distribution to elucidate the response of cells irradiated in an identical manner. A Monte Carlo simulation of GRID irradiation with X-rays implementing an exact replica of the set-up used was used to verify the dose distribution found.

Mapping the dose distribution in cell culture flasks was done using Gafchromic EBT3 films, after first conducting thorough dosimetry of the X-ray irradiation chamber and proton beam used. The EBT3 films were meticulously calibrated before irradiating them inside the cell culture flasks both with and without the GRID collimator. For X-ray irradiation, the average dose received by the EBT3 films in the open field was around 4.6 Gy, whereas when using the GRID collimator, the peak doses (openings under GRID) was around 3.4 Gy, and valley doses (shielded under GRID) around 0.4 Gy. For proton irradiation, the average dose received by the EBT3 films in the open field was 4.7 Gy, the peak dose areas also around 4.7 Gy, and shielded areas around 0.06 Gy. The average dose profile through all films was also found separately for X-rays and protons.

Stained cell colonies in culture flasks irradiated in identically manner to the EBT3 films and a colony counter algorithm were employed. The cell survival found from open field irradiations was used to establish a linear-quadratic (LQ)-model for the cell survival as a function of dose. The LQ-model was subsequently used to make a prediction of the cell survival when irradiating with GRID, by employing it on the dose profiles from the EBT3 films. This was in turn compared to the actual observed survival found from the cell assays conducted with GRID. The observed survival was found to matches with the predicted, except for when irradiating with a nominal dose of 10 Gy using X-rays. Then, the observed survival was significantly lower than predicted in the peak dose areas. The most probable explanation for this – which also explains the huge difference in the peak and valley doses for X-rays compared to protons – is errors associated with the X-ray dosimetry. Since the methodology was established as the experiments progressed, uncertainties were inevitably introduced. However, the X-ray dosimetry matched the Monte Carlo simulations well, where the latter are considered a gold standard for dose calculations. Other attempts to explain the outcome were therefore done, but no definite conclusion could be made as some of the results are partly contradictory.

In conclusion, the methodology for determining the dose distributions with Gafchromic films has been well established, as well as a means for using the dose distribution to analyze the cell survival undergoing various irradiation conditions. This lays a solid foundation for future GRID experiments and is a step toward better understanding the effects of GRID irradiation.

Abbreviations:

AIC	Akaike Information Criterion
AICc	Corrected Akaike Information Criterion
ASMase	Acid Sphingomyelinases
CI	Confidence Interval
CPE:	Charged Particle Equilibrium
CSDA	Continuous Slowing Down Approximation
DNA	Deoxyribonucleic Acid
DSB	Double Strand Break
GNA	Gauss-Netwon Algorithm
GRID	Spatially Fractionated Grid Radiation Therapy
IC	Ionization Chamber
Kerma	Kinetic Energy Released per Mass
LET	Linear Energy Transfer
Linac	Linear Accelerator
LM	Levenberg-Marquardt algorithm
LQ	Linear Quadratic
LRT	Lattice Radiation Therapy
MC	Monitor Chamber
MC	Monte Carlo
MRT	Microbeam Radiation Therapy
MU	Monitor Units
OD	Optical Density
RBE	Relative biological effect
RGB	Red, Green, Blue
ROI	Region of Interest
RSS	Residual Sum of Squares
RT	Radiotherapy
SDD	Source to Detector Distance
SFRT	Spatially Fractionated Radiation Therapy
SSB	Single Strand Breaks
UV	Ultraviolet

Table of contents

1 Introduction

Globally, cancer is the second leading cause of death and it is accountable for 1 of 6 deaths worldwide (WHO). In Norway, around 1/3 of the population will develop cancer during their lifetime with the number increasing as the population grows bigger and older (Kreftregisteret). Cancer is a collective term of around 200 diseases, having in common that abnormal cells grows uncontrollably to a point where they go beyond their usual boundaries, and can then invade adjoining parts of the body and spread to other organs. The goal of cancer treatment is to cure, prolong life, improve quality of life or relieve pain or symptoms. Due to the high number of cancer types and differences in personal health, treatment is specially designed to each person with usual treatment being surgery, radiotherapy or chemotherapy, or combinations of them.

Radiotherapy is a treatment using ionizing radiation, and half of all cancer patients in Norway receive radiotherapy at some point during their treatment (Klepp 2018). In conventional radiotherapy the radiation field is made to cover the entire tumor plus a buffer zone around. This is done to cover potential spread of cancer cells in close proximity of the solid tumor, as well as cover internal movements in the body and inaccuracies in the set-up of the patient. With these large radiation fields, more tissue than just the tumor will inevitably be irradiated. Considering the way X-rays deposit their energy, with a long tail of radiation dose after the maximum point, normal tissue – possibly including sensitive and critical organs – might receive substantial doses of radiation. This can cause severe side effects, which of course is undesirable. One way to tackle this by the use of protons instead of X-rays. Protons have a distinct dose deposition characterized by the "Bragg peak", meaning that most of the dose is delivered at the very end of their track, as opposed to X-rays. This is of great advantage in radiotherapy, as a high dose can reach and span the target while at the same time reducing the dose to normal tissue (Mohan and Grosshans 2017). Two proton therapy centers are to be built in Norway within the next years.

Despite advances in proton therapy, normal tissues will still limit the amount of radiation dose that can be delivered to the tumor. In attempt to overcome this challenge, GRID therapy has been introduced. Here, a spatially fractionated field is used, giving radiation in terms of high-dose spots deposited in systematic pattern over the tumor. This further implies that parts of the tumor will receive little or no dose. Studies have shown that this enables a higher dose and tumor killing while still sparing normal tissue, especially skin. It is also believed to induce radiobiological changes, such as abscopal effects where cell killing is initiated outside of the radiation field, and bystander effects, giving a higher impact than expected to the cells in low dose areas. However, the clinical trials using GRID are few, and several questions regarding their methodology can be raised. The radiobiological mechanisms are not fully understood, and as these mechanisms are likely to be different to those following standard irradiation, the models used for cell survival are not necessarily applicable when using GRID therapy. More research on this is clearly needed to safely bring GRID therapy out in a clinical setting. (Billena and Khan 2019)

The aim of this study was to establish a methodology for studying in vitro effects of GRID irradiation. This was done by obtaining an estimation of deposited dose when irradiating using GRID, both for X-rays and protons, which later could be used to elucidate the survival of cancer cells undergoing the same irradiation conditions. First, thorough dosimetry was done to determine dose rates and field homogeneities of the radiation modalities used. Then Gafchromic EBT3 films, a type a radiochromic film which darken when exposed to radiation, was employed to map the dose distribution inside cell culture flasks in an identical set-up to what was used when irradiating cells. Computer codes were developed to analyze digital scans of the irradiated EBT3 films and obtain profiles of the dose distribution when using GRID irradiation.

Parallelly, experiments with cell colony formation assays were conducted and cell survival data was obtained through a cell counter algorithm. The analysis of the EBT3 films and dose deposition under GRID irradiation could then be seen in comparison with and used to analyze the survival of cells undergoing the same irradiation conditions. Analyses were also done in open fields to establish any potential effects arising from irradiating with GRID specifically.

2 Theory

2.1 Ionizing radiation

The following sections are based on chapter 7 and 8 in “Introduction to Radiological Physics and Radiation Dosimetry” (Attix 1986).

Ionizing radiation is radiation with the ability to excite and ionize matter, i.e. liberate an electron from the atoms of the matter where it interacts. The energy needed to do this is in the range of 4-25 eV and thus, ionizing radiation must carry at least this energy. Ionizing radiation can be divided into directly and indirectly ionizing radiation. Directly ionizing radiation are fast charged particles, directly delivering their energy to matter through many small Coulomb-force interactions along the track of the particle. Indirectly ionizing radiation such photons or neutrons first transfer they energy to charged particles in the matter in a few but large transfers, and these particles then deliver energy to the matter as explained above.

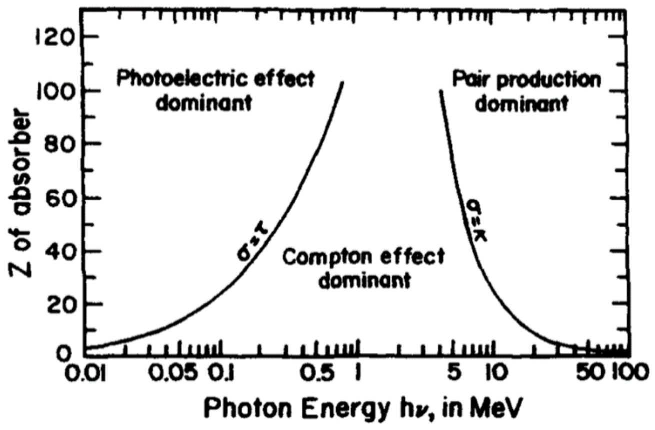


Figure 2.1: Relative importance of the different types of photon interaction. The curves show the interface where the effects have equal likelihood. (Attix 1986)

2.1.1 Photon interactions in matter

There are mainly three types of interaction with matter by photons important in radiological physics. These are photoelectric effect, Compton effect and pair production. Other effects to be mentioned are Rayleigh scatter and photonuclear interactions. However, with Rayleigh scatter there is virtually no energy loss as the photon merely is redirected through a small angle, and photonuclear interactions needs photon energies above a few MeV and occur with low probability, which means they are only interesting with regards to radiation protection due to their production of neutrons. These two effects are therefore not to be discussed in any further details here.

Photoelectric effect

Photoelectric effect is the dominant effect for low photon energies, especially for high Z media, as can be seen in Figure 2.1. When an incoming photon collides with a tightly bound electron it can transfer all of its quantum energy to this electron. This process typically occurs for electrons in the inner shells of an atom with high atomic number. For this to happen, the energy of the incoming photon $h\nu$ must be greater than the binding energy of the electron E_b, $h\nu > E_b$. The electron then receives an energy $T = h\nu - E_b$ and departs at an angle θ relative to the incident photons direction. The atom also departs an angle φ in order to conserve momentum, but the kinetic energy of the atom is essentially zero. A schematic figure of this process can be seen in Figure 2.2.

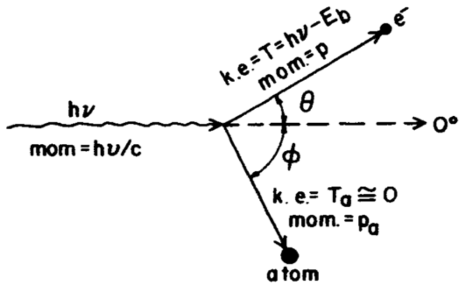


Figure 2.2: Schematic view of the photoelectric effect (Attix 1986).

Compton effect

The Compton effect is the main interaction type for intermediate photon energies and is more important in low Z media (Figure 2.1). The Compton effect happens when an incoming photon hits an electron, assumed stationary and unbound. Some of the quantum energy is transferred to the electron, which departs with resulting kinetic energy at an angle θ relative to the incident photon's direction. The photon is scattered and departs with and angle φ in the same scattering plane, but at the opposite side of the original direction. A schematic figure of this process is shown in Figure 2.3. Momentum and energy are conserved, and the kinematic relations is of course independent of the atomic number of the medium, since we assumed the electron to be unbound.

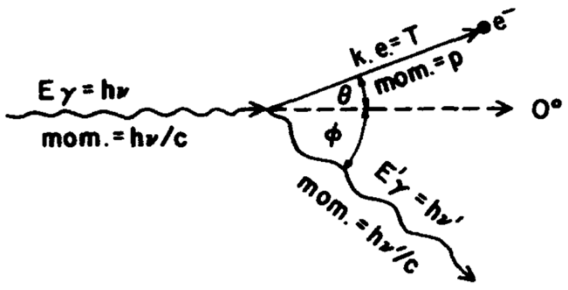


Figure 2.3: Schematic view of the Compton effect (Attix 1986).

Pair production

Pair production dominates for high photon energies and is the creation of an electron and a positron with the disappearance of the photon. This can only occur if the energy of the incoming photon is above a threshold of $2mc^2 = 1.022$ MeV, i.e. the rest mass energy of the particles to be created. Pair production can only happen in a Coulomb force field, typically near an atomic nucleus, but also in the field of an atomic electron. The latter is usually referred to as triplet production, since the atomic electron also will receive enough kinetic energy to be ejected from the site in order to conserve momentum. A schematic figure of pair production in a nuclear coulomb force field is shown in Figure 2.4.

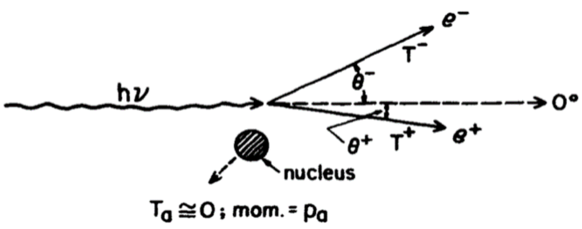


Figure 2.4: Schematic view of pair production (Attix 1986).

2.1.2 Charged-particle interactions in matter

A charged particle cannot pass through matter without interaction, as its Coulomb force field will interact with essentially every atom it passes. This leads to what is called the "continuous slowing-down approximation" (CSDA), a gradual loss of kinetic energy. Where an individual photon only needs a few random interactions to dissipate all its energy, charged particles typically undergo a high number of interactions. This makes it possible to characterize the path length of charged particles by an expectation value, called the CSDA range, R_{CSDA}, as opposed to photons where these predictions are impossible. The type of interactions a charged particle goes through depends on the impact parameter *b,* illustrated in Figure 2.5, and are usually described as soft collisions, hard collisions, and radiative interactions.

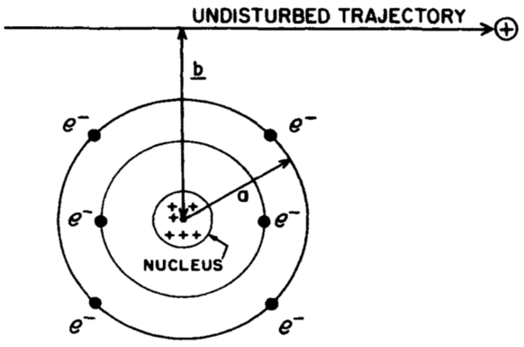


Figure 2.5: Charged particle collision with atom. a is the classical atomic radius, b is the impact parameter (Attix 1986).

Soft collisions, b >> a

Soft collisions are the most common type of interactions, as a large distance from an atom to the incoming particle is the most probable. The large distance makes the whole atom affected by the incoming particle's coulomb force field, resulting in an excitation of the atom to a higher energy level or ejection of a valence shell electron. Either way it results in a very small transfer of energy. Nevertheless, around half of the transferred energy comes from soft collisions, due to their high number.

Hard collisions, b ~ a

Hard collisions are also known as knock on collisions, since it is an interaction between the incoming particle and a single atomic electron. The atomic electron gets a significant transfer of kinetic energy and is ejected from the atom. This electron is called a delta (δ) ray, and creates a separate track of kinetic energy deposits, since the delta ray has enough energy to undergo coulomb force interaction on its own. The fraction of the primary particle’s energy loss from hard and soft collisions are generally comparable, even though soft collisions are far greater in number.

Radiative interactions, b << a

Radiative interactions, also called Coulomb force interactions, occurs when the incoming particle collides with a nucleus, and are mainly important for electrons. In most cases this only deflects the particle without transferring any notable energy to the medium, however, in 2-3% of the cases the collision slows the electron down, resulting in the emission of a photon. These X-ray photons are also known as bremsstrahlung, and for high Z media this is an important mean of energy dissipation.

2.2 Some radiation therapy devices

2.2.1 X-ray beam production and characteristics

X-rays are produced by using an x-ray tube, a vacuum tube with an anode and a cathode having an electrostatic potential difference (typically kilovoltage – kV) between them. The anode and cathode are often made of tungsten. The cathode is heated, causing electrons to be released into the tube by a process called thermionic emission. The potential difference accelerates these electrons to a kinetic energy in the keV region and they then collide with the anode, resulting in the emission of bremsstrahlung. As hard and soft collisions are more prominent, and not lead to bremsstrahlung, most of the kinetic energy is converted into heat. For 10-200 keV electrons as little as 0.1-2% of the total kinetic energy is emitted as bremsstrahlung photons.

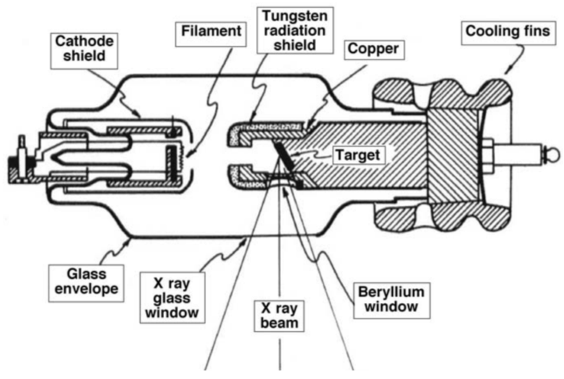


Figure 2.6: Cross section of an x-ray tube. Electrons are liberated from the cathode, and a potential difference between the anode and cathode accelerates them toward the anode. Electrons hitting the anode produces x-ray radiation (Podgorsak 2005).

The energy of the x-ray photons is restricted by the kinetic energy of the accelerated electron, which is given as $T = eV$, where e is the elementary charge and V is the applied voltage over the x-ray tube. The electron normally loses its kinetic energy in a series of interactions but can in rare cases give all to one photon. Thus, the produced photons in an X-ray device come as a continuous spectrum, with the maximum energy of $h\nu_{max} = eV$, as can be seen in Figure 2.7.

When accelerated electrons hit the target, they can knock out orbital electrons in the inner shells. The vacancies are then filled by electrons from higher energy levels, which lead to emission of characteristic photons. These photons will have specific energies corresponding to the energy levels in the atom, and manifests as peaks in the energy spectrum seen in Figure 2.7. These peaks or lines are often referred to as spectral lines and are named after the shell where the atomic electron was residing. , e.g. the K-line, L-line and so forth.

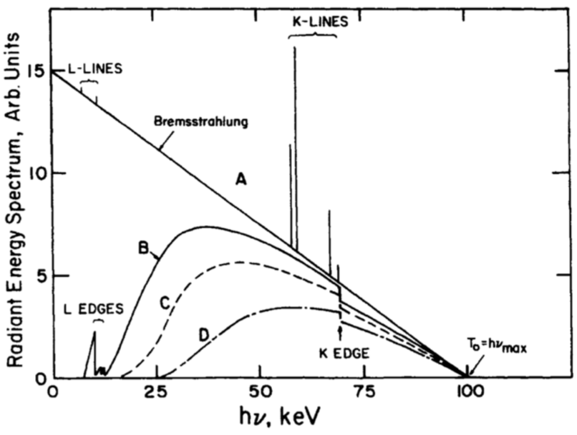


Figure 2.7: Example of an x-ray spectrum with the characteristic K- and L-lines. A shows the unfiltered beam, B, C and D show the same beam with more and more filtering (Attix 1986).

By applying filters to the beam, the photons with lower energies can be removed and the beam becomes less polyenergetic with a higher fraction of high-energy photons. This is called beam hardening, since we are removing the softer, more easily attenuated, photons. The filtering also attenuates the spectral lines, and when using several filters, they should always be placed in descending order of Z, so that each filter can remove the lines originating from the previous filters and smooth the resulting spectrum.

When using x-rays in radiotherapy, higher energies are needed than what is possible from an x-ray tube. This is done using a linear accelerator (linac), accelerating electrons with microwaves before they collide with a metal target and produce high-energy x-rays. More on the mechanisms of linacs below.

2.2.2 Particle beam acceleration

Particle beams are produced by accelerators by first releasing charged particles and then accelerating them, increasing their speed and energy and contain them in well-defined beams using electromagnetic fields. The earliest accelerators used an electrostatic principle, applying a potential voltage difference over a linear tube. The accelerating voltage limits the achievable kinetic energy, like the Cockcroft-Walton generator which can produce energies up to 1 MeV, and Van de Graff generators which go up to 40 MeV (Linder 2020).

Newer accelerators are electrodynamic, using oscillating electric potentials. Their design differs depending on what particles and energies they use. For particles well below the speed of light, like protons, cavities with standing waves separated by areas with no field is used. As the particle speed increases the distance between the cavities becomes greater, allowing the particles to always enter and exit the cavities in phase with the waves and thus accelerating them, as illustrated in Figure 2.8. Protons can be accelerated up to 100 MeV using this technique. Generally, the energy is given by the particles electrical charge q, the applied voltage V, and the number of electrodes N:

$$E = qNV \tag{2.1}$$

For electrons, or other particles reaching velocities closer to the speed of light, a waveguide is used. Here the particles are “carried” by a wave (typically of microwave frequency), with the electrical field vector continuously accelerating them. Electrons can this way be accelerated up to energies of 20 GeV (Helstrup 2019).

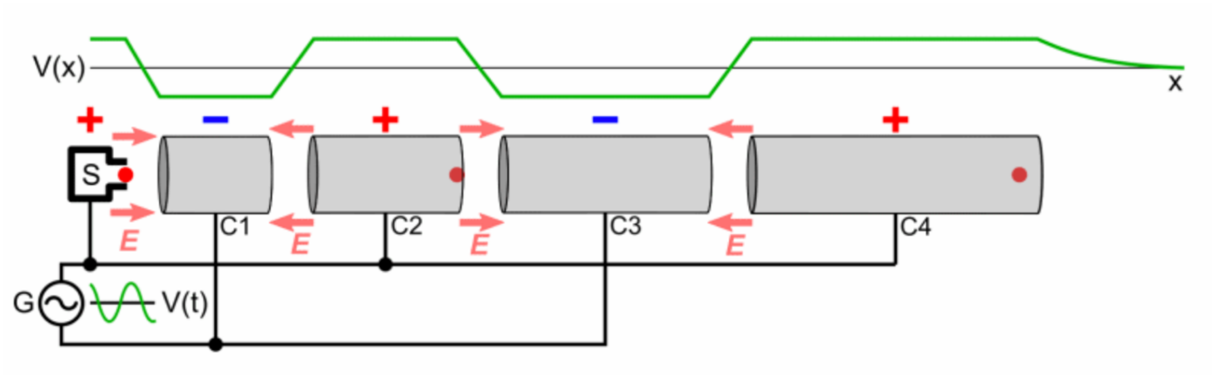


Figure 2.8: Linear accelerator. Particles (red dots) with positive charge originating at source S, accelerated by oscillating electric fields E in gaps between electrodes produced by radiofrequency AC voltage G. V(x) shows the electric potential along the accelerator (Wikimedia Commons 2019).

In addition to linacs, there are cyclic accelerators. The particles can here transit indefinitely, allowing a continuous acceleration and also requiring less space for the same amount of acceleration as a linac. The earliest circular accelerator were cyclotrons, where the particles are accelerated by an electrical field in between two hollow D-shaped plates, called dees. A large dipole magnet bends their path into circular orbits, spiraling outwards as the velocity of the particle increases. A schematic figure of a cyclotron can be seen in Figure 2.9.

Under non-relativistic conditions, the orbital frequency of the particle does not change since the path becomes larger as the velocity increases, and the electrical field shifts with the same frequency. The force acting on the particle in a magnetic field is given by the Lorentz force, and using Newtons 2. law the velocity of the particle can be found:

$$F = qvB = ma = m\frac{v^2}{r} \Rightarrow v = \frac{qBr}{m} \tag{2.2}$$

The frequency needed to accelerate the particle can then be found to only depend on the magnetic field, and mass and charge of the particle:

$$f = \frac{1}{t} = \frac{v}{2\pi r} = \frac{qB}{2\pi m} \tag{2.3}$$

When the particles reach the end of the magnetic field, they are deflected and exits as a beam. The maximum energy of the particles is determined by the strength of the magnetic field and the size of the cyclotron, with ca 10MeV for protons and 40 MeV for alpha particles (Linder.

2018). To further increase the energy, various versions of the cyclotron have been developed, like the microtron or synchrocyclotron. Another variation is the synchrotron, which accelerates particles in a constant radius. Instead the magnetic field is increased as the speed of the particle is increased.

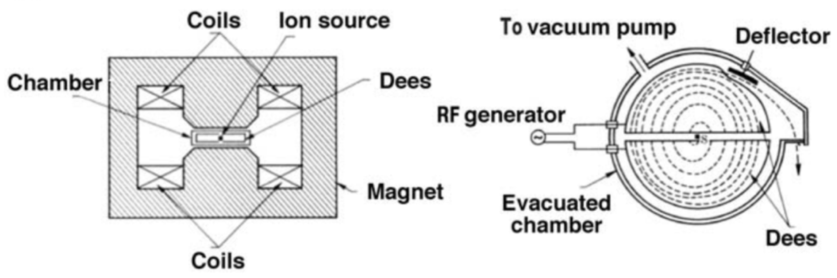


Figure 2.9: Illustration of a cyclotron. The particle is accelerated every time it passes the gap between the two dees. It spirals outwards as the velocity increases until it escapes the accelerator (Podgorsak 2005).

2.3 Dosimetry

Dosimetry is the quantitative determination of energy deposited per mass by directly or indirectly ionizing radiations, in other words how much dose is absorbed by the medium irradiated. This is normally dose by using dosimeters, devices or systems that can measure the dose directly or indirectly. To make this possible, it is necessary to describe the beam through a number of quantities. Some of the most common are defined below, as well as a simplified discussion of cavity theory.

2.3.1 Dosimetric quantities

The following sections are based on Chapter 2 and 8 in “Introduction to Radiological Physics and Radiation Dosimetry” (Attix 1986) and Chapter 2 in “Radiation Oncology Physics: A Handbook for Teachers and Students (Shortt. 2005).

Fluence

Fluence is defined as the expectation value of the number of particles *dN* striking an infinitesimal sphere of cross-sectional area *da*:

$$\Phi = \frac{dN}{da} \tag{2.4}$$

Scattering, absorption, and the creation of new particles will vary the fluence for a radiation field through a medium. Fluence might also vary with time (fluence rate or flux density) and the directional dependence of the energy (differential fluence). The area *da* is always perpendicular to the direction of each incoming particle but by setting a fixed plane, fluence dependent on the angle of the incident beam particles can be obtained (planar fluence).

Energy fluence

Energy fluence takes the energy of the individual rays into account and can be seen as how much energy, *T,* "strikes" the sphere:

$$\Psi = \int_0^{T_{max}} T\,\Phi_T\,dT \tag{2.5}$$

For a monoenergetic beam this reduces to $\Psi = T\Phi$.

KERMA

When indirectly ionizing radiation hits matter the energy is transferred in a two stage prosses, first from photons to secondary charged particles (electrons), then from these particles to the medium. The first step in this energy dissipation is described by KERMA - kinetic energy released per mass. It describes the energy transfer from the indirectly ionizing radiation to the charged particles (electrons), and is defined as:

$$K = \frac{d\varepsilon_{tr}}{dm} \tag{2.6}$$

Where $\varepsilon_{tr} = R_{in,u} - R_{out,u\text{-}rl} + \Sigma Q$, $R_{in,u}$ being the total energy of the photon field entering a volume, while $R_{out,u\text{-}rl}$ is the energy leaving excluding radiative losses from secondary electrons. ΣQ is energy from conversion of rest mass or vice versa. So, KERMA is the expectation value of energy transferred to a volume *dv* with mass *dm*, including radiative-loss energy, but excluding energy transfer between charged particles. ε_{tr} is the total energy transferred from photons to electrons. KERMA can also be expressed in terms of the energy fluence and mass energy transfer coefficient μ_{tr}, and for a monoenergetic beam, this is defined as:

$$K = \Psi\left(\frac{\mu_{tr}}{\rho}\right) \tag{2.7}$$

Kinetic energy given to secondary electrons can further be lost through either collisions or radiative losses, and the total KERMA is usually divided into two parts, $K = K_c + K_r$, where the subscripts refer to "collision" and "radiative" interactions respectively. The collision KERMA is defined as:

$$K_{col} = \frac{d\varepsilon_{tr}^{n}}{dm} \tag{2.8}$$

The net energy transferred ε_{tr}^{n} = $R_{in,u}$ - $R_{out,u}$ + ΣQ, $R_{out,u}$ is all photon energy leaving the volume (including bremsstrahlung), meaning that ε_{tr}^{n} is the total kinetic energy of secondary electrons which is not lost as bremsstrahlung. It can also be expressed in terms of the mass energy absorption coefficient, μ_{en}, and for a monoenergetic beam this is:

$$K_{col} = \Psi\left(\frac{\mu_{en}}{\rho}\right) \tag{2.9}$$

The collision KERMA can also be written as $K_{col} = K(1 - g)$, with g being the bremsstrahlung fraction.

Absorbed dose

The definition of absorbed dose is the expectation value of the energy imparted to matter per unit mass at a point:

$$D = \left(\frac{d\varepsilon}{dm}\right) \tag{2.10}$$

Here, ε = $R_{in,u}$ + $R_{in,c}$ - $R_{out,u}$ - $R_{out,c}$ + ΣQ, and $R_{in,c}$, $R_{out,c}$ is the energy respectively entering and leaving the volume coming from charged particles. Under the presence of charged particle equilibrium (CPE) where the number of charged particles entering the volume is the same as leaving, $R_{in,c}$ = $R_{out,c}$, and the energy imparted reduces to the net energy transferred, the absorbed dose equals the collision KERMA, equation (2.9). This can be seen in Figure 2.10.

A more realistic situation, however, is to have a region of transient charged particle equilibrium (TCPE) due to scattering in the medium and photon attenuation, where electrons originating from upstream contributes to the dose, while the collision KERMA gives the photon contribution $R_{in,u}$ - $R_{out,u}$. The absorbed dose is then practically proportional to the collision KERMA, as illustrated in Figure 2.10, since the range of generated electrons does not change appreciably with depth in the medium.

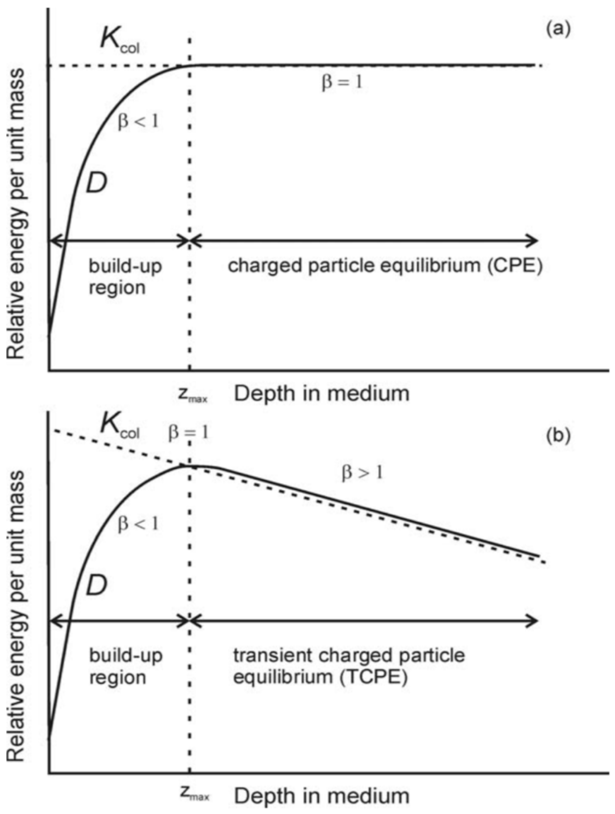


Figure 2.10: Absorbed dose and collision KERMA as a function of depth when irradiated by a high-energy photon beam. (a) is the hypothetical case of no photon attenuation or scattering, and (b) the realistic case (Shortt. 2005).

Stopping power

When charged particles interact with matter, the force slowing them down is referred to as stopping power. It is the expectation value of the rate of energy loss per unit of path length x, (dT/dx). By dividing stopping power by the density of the medium, the mass stopping power ($dT/\rho dx$) is obtained. Stopping power can be divided into collision stopping power and radiative stopping power, coming from bremsstrahlung or in rare cases other radiative processes, $(dT/\rho dx) = (dT/\rho dx)_c + (dT/\rho dx)_r$, where the subscripts denote collision and radiative respectively. Energy spent in collisions contributes to the dose near the track, while radiative losses carries the energy away. The mass collision stopping power can be further divided in two, with one component originating from hard collisions and one from soft collisions, $(dT/\rho dx)_c = (dT_h/\rho dx)_c + (dT_s/\rho dx)_c$. The soft collision term was derived by Bethe, using the Born approximation, and is valid for all charged particles. The hard collision term, however, differs with the different type of incoming particle. Combining the soft and hard term with a few corrections, the mass collision stopping power for heavy charged particles is given:

$$\left(\frac{dT}{\rho dx}\right)_c = 0.3071\ \frac{Zz^2}{A\beta^2}\left[13.8373 + \ln\left(\frac{\beta^2}{1-\beta^2}\right) - \beta^2 - \ln I - \frac{Z}{C}\right] \tag{2.11}$$

Units for stopping power as given above is MeV cm^2/g. *Z/A* is the number of protons per nucleon, *z* is the particle charge, $\beta = v/c$ and is related to the kinetic energy of the particle, *I* is the mean excitation potential of the atom, and *Z*/*C* is a shell correction significant when β is small. This gives the characteristic Bragg-peak for e.g. protons – the particle deposit little dose until the velocity is low, then the remaining energy is almost deposited at once. This is illustrated in Figure 2.11.

For electrons the hard collision term is based on the Møller cross section, and for positrons the Bhabha cross section. Combining this with Bethes soft collision formula, gives equation (2.12), where $\tau \equiv T/m_0c^2$, $F^{\pm}(\tau)$ is a function but will not be further discussed, and δ is a density correction needed since electrons (and positrons) may polarizes the medium being traversed (this is negligible for heavy charged particles).

$$\left(\frac{dT}{\rho dx}\right)_c = k\left[\ln\left(\frac{\tau^2(\tau+2)}{2(I/m_0c^2)^2}\right) + F^{\pm}(\tau) - \delta - \frac{2C}{Z}\right] \tag{2.12}$$

The mass radiative stopping power is insignificant for heavy charged particles, and the mass stopping power is the same as the mass collision stopping power almost exactly. For electrons and positrons, the mass radiative stopping power is the rate of the energy loss resulting in production of bremsstrahlung, and is based on the Bethe-Heitler theory. I will not go into further details here.

In order to calculate the energy transferred to a region of interest, the concept of restricted stopping power has been introduced. δ-rays from hard collisions might carry energy far away from the track of the primary particle, but by introducing a cut-off value Δ they are not included as 'local' energy losses. This way, overestimation of the dose to an area is avoided. The restricted collision stopping power is also known as *linear energy transfer (LET)*, L_Δ, and is the mean energy loss per unit length $L_\Delta = (dT/\rho dx)_\Delta$. If the cut off energy is equal to T_{max}, we get the unrestricted linear energy transfer L_∞, which equals the stopping power.

Examples of depth dose curves for both protons, electrons and photons are shown in Figure 2.11. Photons have an initial build-up before quickly reaching a peak dose which is followed by a long tail with gradually decreasing dose. Electrons deliver all dose very close to the entrance, with a steep fall and zero dose behind this, while protons have a low entrance dose which is only slightly increasing until reaching a point where almost all dose is delivered at once, before quickly falling to zero.

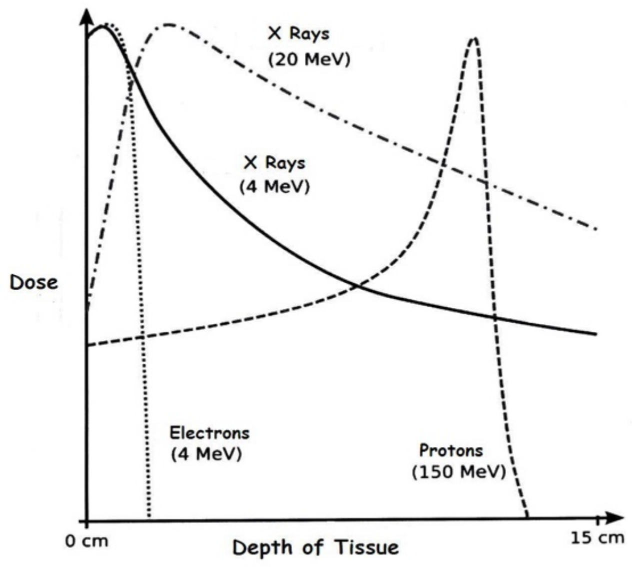


Figure 2.11: Depth dose curves for x-rays, electrons and protons entering tissue (Wikimedia Commons 2018).

Exposure

When doing dosimetric measurements, it is often the exposure that is measured. Exposure X is the number of charges of one sign produced in a gas of mass m, and is a measurement of the amount of radiation going through the air. It is only defined for X-rays and photons, with. the definition being:

$$X = \frac{dQ}{dm} \tag{2.13}$$

Exposure is the ionization equivalent of collision KERMA in air, and they are related to each other as:

$$X = \frac{e}{W} K_{col,air} = \frac{e}{W} K_{air}(1 - g) \tag{2.14}$$

With W being the mean energy required to produce one ion pair, and e the electron charge.

2.3.2 *Cavity Theory*

The following sections are based on Chapter 10 in "Introduction to Radiological Physics and Radiation Dosimetry" (Attix 1986) and Chapter 2 in "Radiation Oncology Physics: A Handbook for Teachers and Students (Shortt. 2005).

The definition of dose is only valid in a given medium, as two different media in the same radiation field not receives the same dose. When measuring absorbed dose, and the sensitive medium of the dosimeter is most often different from the medium where the dose estimate is needed. Cavity theory tries to relate the dose received by the sensitive medium of the dosimeter, also called cavity, to the dose absorbed by the surrounding medium. There are various cavity theories, depending on the size of the cavity.

Bragg-Gray cavity theory

Bragg-Gray cavity theory is valid for small cavities, as it assumes the cavity does not disturb the fluence of charged particles in the medium. This can only be valid in regions of CPE or TCPE, implying that the range of the charged particles incident must be large compared to the cavity. It also assumes photon interaction to be negligible, and only considers the crossing charged particles to deposit dose. This relates the dose to the cavity to the dose in the medium as shown in equation (2.15), where $(S/\rho)_{med}^{cav}$ is the ratio of the mass collision stopping power of the two media.

$$\frac{D_{cav}}{D_{med}} = \left(\frac{S}{\rho}\right)_{med}^{cav} \tag{2.15}$$

The Bragg-Gray theory has been extended in several ways, like the Bragg-Gray Laurence theory which includes the slowing down spectrum of particles generated in the wall. Spencer and Attix also derived a more general formulation of the Bragg-Gray theory, known as Spencer-Attix cavity theory. Here, the creation of δ-rays is taken into account, as some of them are energetic enough to escape the cavity, and do not deposit dose there. The stopping power in equation (2.15) therefore needs modifications. Even with this, real applications of Spence-Attix theory need more correction factors, e.g. depth corrections. The latter method is also often referred to as stopping power theory since eq 2.15 is the central element, and it plays an important role in determining both relative and absolute dose.

Charged particle equilibrium (CPE)

For large cavities the dose contribution from charged particles outside the cavity is insignificant when compared to the contribution of the electrons created inside. The dose ratio of the cavity vs medium then equals the collision KERMA ratio, and from equation (2.9) therefore also the ratio of the average mass absorption coefficient of the cavity to the medium, $\left(\mu_{en}/\rho\right)_{med}^{cav}$. The dose to the medium is then related to the cavity dose as:

$$\frac{D_{cav}}{D_{med}} = \left(\frac{\mu_{en}}{\rho}\right)_{med}^{cav} \tag{2.16}$$

Burlin cavity theory

For intermediate sized cavities, neither of the theories above is appropriate. To fill the gap, Burlin cavity theory was created. It assumes that (1) the wall and cavity to be homogenous, (2) a homogenous photon field existing throughout the medium and cavity, (3) CPE exists at all points except within ±one electron range from the interface between the medium and cavity, and (4) for secondary electrons generated in the cavity and medium the equilibrium spectra is the same.

$$\frac{D_{cav}}{D_{med}} = d\left(\frac{S}{\rho}\right)_{med}^{cav} (1 - d)\left(\frac{\mu_{en}}{\rho}\right)_{med}^{cav} \tag{2.17}$$

Burlin's dose equation can be seen in equation (2.17). It is a weighted mean of the stopping power and CPE-base theories (eqs. 2.15 and 2.16), with the weighing factor *d* being the average attenuation of electrons generated in the crossing of the cavity. This makes the formula approach Bragg-Gray theory for smaller cavities and CPE theory for larger. *d* can be found mathematically, but I will not go into further details. Burlin cavity theory has been shown to work adequately for some cavities, but may need more corrections, and is not used extensively today.

2.3.3 Radiation dosimeters

The following sections are based on chapter 12 and 13 in "Fundamentals of Ionizing Radiation Dosimetry" (Andreo. et al. 2017).

Dosimeters are devises used to measure dose, and a normal configuration for this is a sensitive volume surrounded by some sort of wall. The wall is not only a mechanical protection of the sensitive volume, but also acts as a source of secondary electrons, shields the volume from electrons origination outside the wall, and can filter the beam if desired. There are both absolute and relative dosimeters, where the latter requires calibration and only gives a reading proportional to the dose, whereas an absolute dosimeter directly provides a measure of absorbed dose.

Ionization chambers

Ionization chambers are based on the principle of ionometry – the science of measuring ionizations. They consist of two electrodes with a gas chamber in between. There are four main types of ionization chambers: cavity chambers, parallel-plate chambers, transmission chambers and free-air chambers. The latter two will not be discussed here.

By applying a voltage to either the wall or the collecting electrode, typically 200 V to 400 V, the gas in the chamber is in an electric field. The incident radiation will then ionize this gas, liberating electrons that will be collected at the positive electrode. This induce a current that is measured by an electrometer which is connected to the collection electrode, with the measured current being proportional to the number of ionizations. The chamber response will vary with temperature, pressure air humidity and also the applied potential. Every ionization chamber used for dosimetry is calibrated, and this happens under certain reference conditions: air temperature $T_{ref} = 20°C$, air pressure $P_{ref} = 101.325$ kPa and relative humidity $h_{ref} = 50\%$, and a stated applied potential. If the chamber is to be used under different conditions, this must be calibrated for by using correction factors.

One of the most used cavity chambers is that in the shape of a cylinder or thimble. Figure 2.12 shows a cross section of this chamber type. It has a coaxially positioned internal anode surrounded by a cylinder-shaped, metal coated wall that makes up the electrodes. The cavity in between is filled with gas. Manufacturing imperfection might make the chamber sensitive to orientation, and it is recommended to always use the same orientation as at calibration. The required wall thickness depends on the application, if used in a photon beam in free air the wall should be thick enough to ensure CPE, in which case the Bragg-Gray theory can be applied. In phantoms a thin wall approach is more common because it enables the use of the stopping power ratio with a perturbation correction. A common solution is a thin wall with a thicker cap or shell that can be applied when needed.

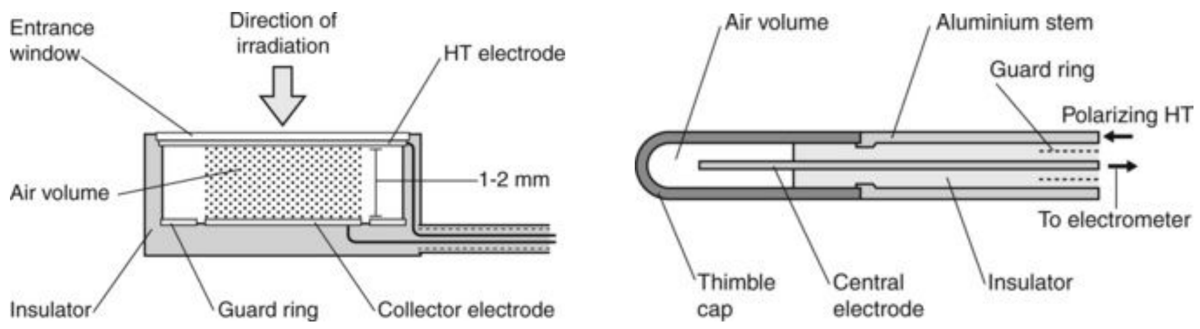


Figure 2.12 Ionization chambers. ***Left:*** *Parallel plate chamber. Collector electrode collects ionization produced within the shaded air volume.* ***Right:*** *Thimble chamber. Central electrode collects ionization produced within the air volume. (Radiology Key 2017)*

Parallel plate chambers are made up of two circular electrodes separated by a small gap of around 2 mm or less, as seen in Figure 2.12. This is in reality a type of cavity chamber, but with an extremely thin entrance disc they are more suitable for low photon energies and electron beams. The thin entrance also allows for better depth-dose measurements with good spatial resolution including more accurate measurements near the surface. However, it also makes them less stable in the long term, and, depending on the thickness of the collector and plate separation they might suffer from a notable polarity effect.

Radiochromic film

There are other ways to measure dose besides ionization chambers. Radiochromic film is a chemical dosimeter that changes its optical characteristics when exposed to radiation. It is made of a polyester base and a gelatine matrix containing a radiation sensitive monomer. This reacts to the radiation by polymerizing diacetylene molecules, forming blue polydiacetylene dye polymers which absorb light from the red part of the visible spectrum. These molecules are either rectangular, or for the Gafchromic EBT emulsions, in the form of tiny needles. Because of the small size of the formed molecules, it gives a high spatial resolution with up to 1200 lines per mm.

More dose gives more opacity to the film. Following irradiation, this color change happens without any processing of the film and is immediately visible to the naked eye, as illustrated in Figure 2.13. The true response is obtained by measuring the change in optical density (OD) before and after irradiation, normally by a light transmission or reflection measurement. However, measured OD depends on the wavelength at which the transmission is done. This means that for radiochromic film dosimetry, the response curve will be unique depending on type of film, type of film scanner and dosimetry protocol. Individual batch calibrations should therefore always be done, even for films of the same type, as there can be small differences between different batches. The newest film generations are more insensitive to light but should still be kept in the dark as much as possible, especially avoiding sun light as UV rays might darken the films. The films are also independent of radiation energy and dose rate, although some films might have ranges of energies where they work more optimally.

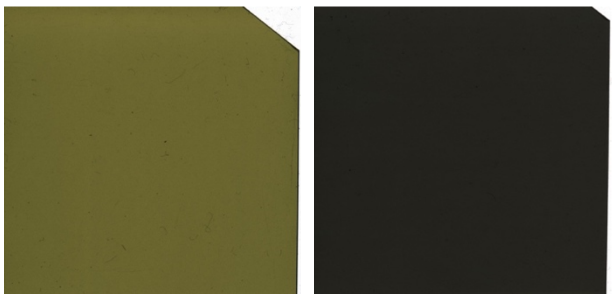


Figure 2.13: Gafchromic EBT3 films. Unirradiated to the left, irradiated with 10 Gy to the right.

2.4 Radiobiology

2.4.1 Cell cycle and checkpoints

This section is based on chapter 17 and 18 in Molecular Biology of the Cell (Alberts. et al. 2015).

Cells go through the cell cycle with the purpose to create new cells, by duplicating all components of the cell and split into two genetically identical daughter cells. The cell cycle is normally divided into four phases: G1, S, G2 and mitosis. S-phase is where the DNA replication and chromosome duplication happen, and for a typical mammalian cell this occupies half of the cell-cycle time. In mitosis, the actual cell division takes place. In between there are two gap-phases G1 and G2, where the cell has time to grow and duplicate the other cell components. All phases, particularly G1 and G2, contain important checkpoints for monitoring whether the cell is ready to go into new stages of the cell cycle. One of the major checkpoints is in G1 where it is decided whether the cell should continue in the cell cycle. The DNA has to be intact and the cell have to be ready to start DNA replication in order to proceed, and it is also determined whether the organism need to produce more of these cells. If not, the cell is sent to a resting phase, known as G0.

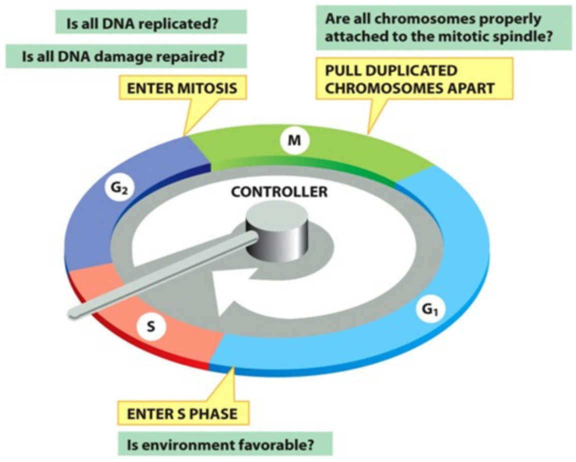

Is all DNA replicated?
Is all DNA damage repaired?
ENTER MITOSIS
Are all chromosomes properly attached to the mitotic spindle?
PULL DUPLICATED CHROMOSOMES APART
M
G_2
CONTROLLER
G_1
S
ENTER S PHASE
Is environment favorable?

Figure 2.14: The four phases of the cell cycle with the control system represented as a rotating arm. The control system triggers specific processes when it reaches the different transition points, and might also arrest the cell there if necessary (Alberts. et al. 2015).

Another major checkpoint is found in G2, where it is checked if the DNA replication was successful and is finished before the cell can enter mitosis. S-phase is often presented as a checkpoint in itself, as the DNA replication is controlled consecutively. There is also a checkpoint in the resting phase G0, to make sure the cell is ready and that the surroundings allow the cell to leave the resting phase and start the cell cycle. Mitosis also has a checkpoint right before the cell division actually happens. If any problems in any of the checkpoints are detected, the cell will be held back until this is solved. An overview if the phases and control is shown in Figure 2.14.

Loss of checkpoint control is a common feature in cancer cells, and often comes from a malfunction in one or more of the regulatory proteins involved at the checkpoints. Genes bringing cells past the G0 checkpoint, starting the cell cycle, are called proliferation genes. A mutation in one of these genes can lead them to act without listening to signals from the body.

This way, they start sending cells to the cell cycle even when they are not needed, and this way a cancer – which is cells growing out of control – can start to develop. On the opposite side are tumor suppressor genes. These are genes preventing cells from entering cell cycle, e.g. by recognizing DNA damage and therefore stopping the cell from entering S-phase. If these genes are not expressed correctly, cells that should have been stopped from entering or progressing through the cell cycle will continue, which also can lead to the development of cancer. In fact, 50 % of human cancers carry a mutation in a tumor suppressor gene known as *p53* (Ozaki and Nakagawara 2011).

2.4.2 Radiation induced damage and repair systems

This section is based on Chapter 2 in "Radiobiology for the Radiobiologist" (Hall and Giaccia 2019).

The DNA is the genetic material of the cell and is of special interest here as it is commonly accepted that this is the radiosensitive part of the cell. The DNA has a double helical structure with two DNA strands with a backbone of sugar and phosphate groups. Four different bases are attached to this backbone, and the strands are held together by hydrogen bonds between these bases. The bases are complementary: adenine and thymine pairs, and guanine and cytosine pairs. Adenine and guanine are purines, double ring groups, while cytosine and guanin are pyrimidines, single-ring groups. An illustration of a DNA is shown in Figure 2.15.

The different base sequences make up the genetic codes, although most of the human DNA is noncoding. The DNA is tightly packed around histones, forming chromosomes. During replication of DNA, the strands are split from each other, and each of them act as a template for a new strand. The replication process is fast, and errors inevitably happens. However, the body has a solid system for detecting and correcting these errors, with over 99.9% of the errors being repaired without flaws. Compared to the number of errors happening naturally during the replication process, the number of damages caused by radiation is small. Nevertheless, radiation can induce a number of different damages to the DNA and induce several different repair processes.

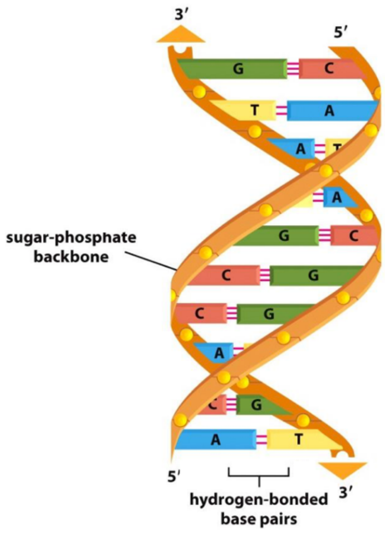


Figure 2.15: DNA. Two complementary strands twist around and form a double helix. The strands are made of four different bases attached to a sugar-phosphate backbone. The two strands are held together by hydrogen bonds (Alberts. et al. 2015).

Radiation induced damage breaking one of the stands in the DNA are called single strand breaks (SSB). These are readily repaired using the opposite DNA strand as a template, and therefore not of high biological importance when it comes to cell killing – apart from the fact that a misrepair can lead to mutations. However, two SSBs close enough in time and space can lead to a double stand break (DSB), which also can also be made directly by single tracks of radiation. The repair of a DSB is more intricate and will also depend on where in the cell cycle the cell is located. Radiation can also induce base damage. Even though base damage happens spontaneously during DNA replication, the rate of the damages increases when exposed to radiation. The most frequent spontaneous damages are depurination; detachment of a purine from the sugar-phosphate backbone, and deamination; when a base spontaneously transform into uracil, the base replacing thymidine in RNA. The different types of damage to a cell are often called lethal, potential lethal or sublethal damage. A lethal damage is a damage that will kill the cell, with no hope for repair. A potential lethal damage can be repaired, but if this not happens in time the cell will die, like after a DSB. Sublethal damages are not deadly in themselves but can be if combined with other sublethal damages, like two SSB turning into a DSB.

The danger of damages to the cell is not only cell death, and (unless numerous cells dies) an organism can ordinarily cope with losing some cells. It is more critical if cells survive with severe mutations to their DNA in such a way that they no longer respond to the signals from the body, as this might initiate cancer. Of all damages, DSB is the most critical as it both can lead to cell death and mutations, particularly chromosome aberrations. The broken ends originating from a DBS are "sticky" and can rejoin with other "sticky" ends. This can result in a restauration of the break, but if the sticky ends reattach to other broken ends the result is highly distorted chromosomes. The breaks might also fail to rejoin at all, causing deletion of the chromosome. Examples of lethal aberrations are dicentrics, rings and anaphase bridges, where the latter are chromatid aberrations. Two other important aberrations are symmetric translocations and small interstitial deletions. Although not lethal, they are mutations which frequently are involved in the development of cancer. An overview of some typical chromosome aberration caused by radiation is shown in Figure 2.16.

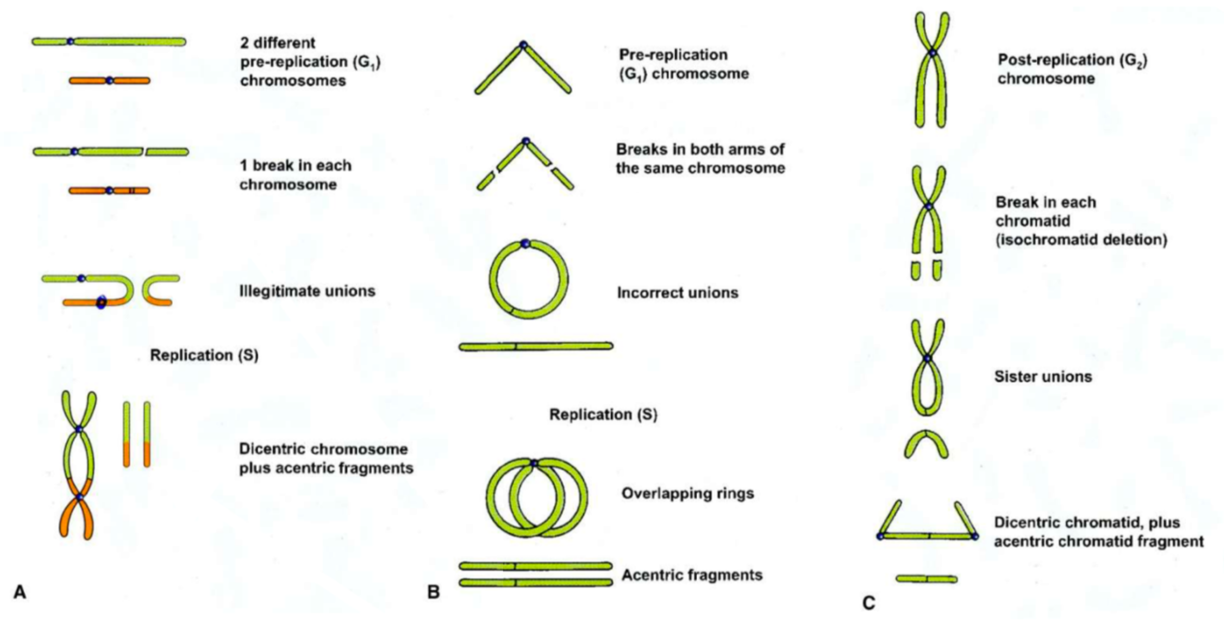

2 different pre-replication (G_1) chromosomes
1 break in each chromosome
Illegitimate unions
Replication (S)
Dicentric chromosome plus acentric fragments
A
Pre-replication (G_1) chromosome
Breaks in both arms of the same chromosome
Incorrect unions
Replication (S)
Overlapping rings
Acentric fragments
B
Post-replication (G_2) chromosome
Break in each chromatid (isochromatid deletion)
Sister unions
Dicentric chromatid, plus acentric chromatid fragment
C

Figure 2.16: The steps in the formation of different chromosome aberrations. ***A:*** *Dicentric.* ***B:*** *Ring.* ***C:*** *Anaphase bridge. (Hall and Giaccia 2019)*

As there are several types of damages to the DNA, there are also several repair processes. The most complete reparation of a DSB is called homologous end-joining (HEJ). In G2 phase, after the replication of DNA, two chromosomes in each nucleus are bound together. In a DSB, some base pairs are usually lost, but if the cell is in G2-phase, the extra chromosome can be used as a template for the broken DNA and replace the lost information. If the cell is in G1-phase on the other hand, no such template is available, and the most common repair is a nonhomologous end-joining (NHEJ). Here the broken ends are fused together without consideration for lost information. This is an error prone reparation, but since most of the DNA is noncoding, it can be worth the risk of making the DNA strand intact, even with loss of bases.

Base damages like depurination and deamination are repaired by the base excision repair (BER). This works when only one strand has damage while the other strand still is intact, and the damaged bases are removed and replaced before the strand is sealed. Pyrimidine dimers or other bulky adducts are repaired by the nucleotide excision repair (NER), where the essential steps are to recognize damage, bracket the lesion through DNA incisions and remove the region with lesions, before repairing, filling out the gap and sealing the DNA. A base-base or small insertion mismatch in the DNA is fixed by the mismatch repair (MRR). Here, the mismatch is detected, the incorrect nucleotides are cut off, the gap is filled, and the DNA is finally sealed.

Other types of DNA damage worth mentioning are crosslinks, either on one strand or between two, called intrastrand and interstrand crosslinks respectively. This is problematic as it blocks DNA polymerase activity during the DNA replication and can lead to cell death. The intrastrand crosslinks are usually dealt with by DNA polymerase switching to a strand without the crosslink. This is not possible with interstrand crosslinks, which is solved by NER or HEJ depending on the location in the cell cycle.

2.4.3 Clonogenic survival

This section is based on Chapter 3 in "Radiobiology for the Radiobiologist" (Hall and Giaccia 2019).

A clonogenic cell is a cell with the ability to divide indefinitely and produce colonies, where a colony normally is defined as a group of 50 cells or more. Establishing how different cells respond to radiation in terms of their clonogenic ability is normally done in vitro (i.e. in a culture dish). Typically derived from tumor tissue in humans or rodents, tiny pieces of tissue are taken and dissolved in a trypsin suspension which loosens the cell membrane. The cells are then seeded onto a dish with an appropriate growth medium, and if they are kept at 37°C and aseptic conditions, they may attach to the surface, grow and divide. Every few days, the cells are removed and the dish recultured with a small number of cells which then repopulates the dish.

Normally, not all cells in a dish produces colonies, due to factors like stress of manipulation, suboptimal growth conditions or errors in the counting. Plating efficiency is a term indicating how many of the seeded cells actually grow into colonies, even without any outer influence. It is given as the number of counted colonies divided by the number of plated cells. When looking at the survival of irradiated cells, the plating efficiency have to be taken into account as not all cells would have grown into colonies even without radiation. The surviving fraction is then given as the colonies counted divided by the cells seeded and the plating efficiency:

$$\mathrm{PE} = \frac{\text{colonies counted}}{\text{cells plated}} \tag{2.18}$$

$$\mathrm{SF} = \frac{\text{colonies counted}}{\text{cells seeded} \cdot \mathrm{PE}} \tag{2.19}$$

When establishing survival curves for cells, survival and/or death have to be defined. A cell can for example be considered dead if it has lost a specific function or the inability to continue proliferation. Even if a cell is intact and can produce proteins – maybe even go through the cell cycle a couple of times – it can still be defined as dead if it has lost the ability to divide indefinitely. This is true for radiotherapy, where the goal only is to kill the cells in the sense that they no longer can divide, and therefore not contribute to further growth or spreading of a malignancy.

2.4.4 LQ-modelling

This section is based on Chapter 2 in "Molecular Radiation Biology" (Dertinger and Jung 1970) and Chapter 2 and 3 in "Radiobiology for the Radiobiologist" (Hall and Giaccia 2019).
The linear quadratic model (LQ model) describes the probability for a cell to survive when exposed to a dose of radiation. The LQ model is based on that cell survival is linked to damage of the DNA. There is a probability α for having a lethal damage, like a DNA DSB coming from a single particle. This probability increases linearly with dose. There is also a probability β coming from the cumulation of sublethal damages, like two DNA SSB from two different particles becoming a DSB if close enough in time and space. The creation of a DSB is thus time dependent, as an SSB has half-life of 15 min. This way, repair is included in the model since a

quick repair of an SSB not will result in a DSB. In addition, the distance between the breaks must be small enough to result in a DSB. High doses will accumulate a vast amount of SSB, making the cells unable to repair all. This results in the β term scaling as a square of the dose, making it dominate for high doses.

Combined, the probability for having a lethal DSB is $q = \alpha D + \beta D^2$. In a single hit, single target system, the probability of having an unrepaired DSB leading to cell inactivation is found using a Poisson distribution,

$$p(n) = \frac{q^n}{n!} e^{-q} \tag{2.20}$$

where n is the number of hits required to inactivate the target. Hence, it keeps its functionality with n-1 hits or less, and the probability for this is then

$$p_{func} = e^{-q} \sum_{k=0}^{n-1} \frac{q^k}{k!} \tag{2.21}$$

In our case, the cell is of course the target, and one hit is enough to cause cell death – through a DSB. If N_0 cells are irradiated, and N is the number of cells not inactivated, we end up with

$$S = \frac{N}{N_0} = e^{-q} = e^{-\alpha D - \beta D^2} \tag{2.22}$$

α and β then describes the radiosensitivity of the cell, and D is the delivered dose.

A typical cell survival response is shown in Figure 2.18. The dose where the contribution from the linear (αD) and quadratic (βD^2) terms are equal gives the α/β ratio. This ratio indicates how curved the response is, as a high α/β gives a straighter curve, while a low α/β gives more curvature. This becomes particularly significant if the dose delivered is separated into a series of equal fractions, since β represent sublethal damage. A low α/β ratio indicates a higher β and thus more sublethal damage than cells with a high α/β ratio. If the time between the fractions are sufficiently long for cells to repair this sublethal damage, they will respond to the next dose as if they had received no dose beforehand, repeating the initial "shoulder" of the curve, as illustrated in Figure 2.17. This effect is critical in clinical radiotherapy, as tumors and normal tissues have different α/β, enabling a high relative level of cell killing for tumors compared to normal tissues when irradiating with a fractionated regime.

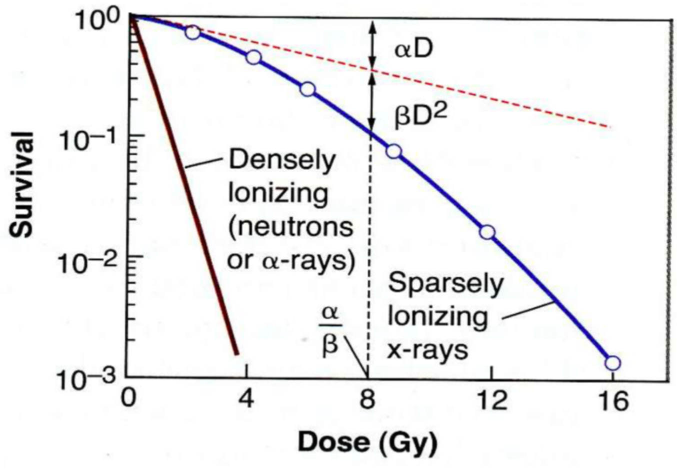


Figure 2.18: Example of an LQ-curve, both for densely and sparsely ionizing radiation (Hall and Giaccia 2019).

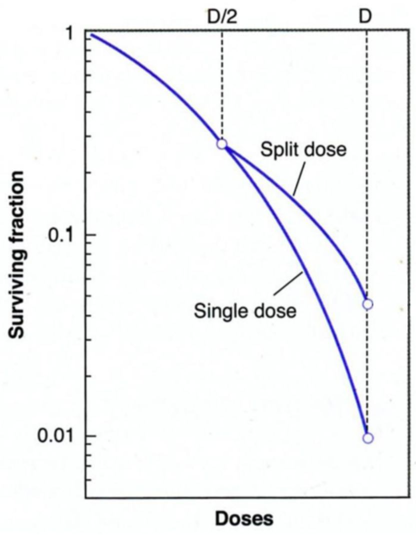


Figure 2.17: Single dose vs the same total dose delivered over 2 fractions. The surviving fraction is notably higher when using the fractionated regime, since the shoulder of the curve is expressed for each fraction of dose (Hall and Giaccia 2019).

For seven or more decades of cell killing, the survival curves found becomes straight, and not continuously bending as the LQ model describes. However, the LQ model is still an adequate representation of data for the doses used in clinical radiotherapy, and also has the advantage of only using two adjustable parameters.

2.4.5 *Radiation-induced bystander effects*

This section is based on the papers «Bystander effects and radiotherapy” (Marín et al. 2014) and “The Bystander effect” (Hall 2003).

Bystander is an effect where unirradiated cells in proximity of irradiated cells shows more biological damage than expected, for instance DNA damage, chromosomal instability, mutation and apoptosis. There are several mechanisms behind this effect, one is intercellular communication such as gap junction mediated signals, where the cells have to be in contact in order to transmit signals. Another mechanism is medium mediated signaling, where soluble factors are released through the medium and can be transported to more distant cells. It is however not clear whether these two different signaling processes are in fact the same phenomena, even though both are called bystander effect. The bystander effect varies with cell lines and dose received, as high doses are dominated by direct effects (e.g. DSB), where the bystander effect saturates.

Bystander effects can be both damaging and protecting, by the release of either cytotoxic signals or by promoting growth factors and inhibiting toxic factors. The effect is largest when the cells are in close contact, i.e. through gap junction signaling, and it has been shown that up to 30% of bystander cells can be killed by this mechanism if the cells are situated close enough. This means that the target for radiation damage actually is larger than the cell itself, which also has been showed in vivo where partial organ irradiation has induced out-of-field effects.

The term bystander effect is also used about an effect seen when the medium of irradiated cells is transferred to unirradiated cells. This too will induce biologic effects and indicates that cells release factors into the medium capable of killing unirradiated cells. Medium irradiated without cells show no such effect. However, not all cells are capable of either producing or receiving these factors, and there is a great variation in the release of bystander factors. The data regarding bystander effects mainly comes from in vitro studies, and the biological significance of this effect in vivo is not fully known.

2.4.6 Abscopal effects

This section is based on the paper “Abscopal effect of radiotherapy combined with immune checkpoint inhibitors” (Liu et al. 2018).

The word abscopal loosely means “off-target” and is used for the effect seen in radiotherapy when irradiating one area gives a tumor response in an unirradiated area. The influence of radiotherapy (RT) on the tumor microenvironment is complex as localized radiation gives rise to a set of opposing mechanisms, like the release of a variety of both pro- and anti-immunogenic cytokines and chemokines. One example is interferons which is one of the main contributors to the therapeutic effect of RT, while other factors like TGFβ acts immunosuppressive and also causes radioresistance of the tumor. In addition, a variety of signal molecules are released, also having dual effects on the tumor and tumor environment.

The abscopal effect is believed to be induced by immunogenetic cell death. This is a special type of cell death promoted by RT, where irradiation causes subtle changes in the dying tumor cells. The protein calreticulin (CRT), acting as an “eat-me” signal, may be promoted to the cell surface, and several damage-associated molecular patterns (DAMP), like ATP and a protein called HMGB1, could be released. HMGB1 acts as a danger signal, while ATP is a “find-me” signal resulting in the secretion of pro-inflammatory cytokines. These and other effects interact with dendritic cells which in turn migrates to lymph nodes, where the antigens are presented to T cells. Combined with other activation signals, T cells – especially CD8+ T cells – are triggered, and by circulating through the blood stream they are able to destroy cancer cells in unirradiated parts of the body.

Although this is an interesting and potentially beneficial effect, it is not fully understood. Also, the clinical significance remains scarce with only 46 reported cases from 1969 to 2014. RT induces a variety of complex effects and is alone mainly thought to be immunosuppressive, counteracting the abscopal effect. Studies have shown that a combination of RT and immunotherapy might overcome this, as immunotherapy enhances the systemic anti-tumor effects of RT. In the end, this might be thus used as an in-situ vaccine. The abscopal effect is also dependent of dose and fractionation regime, and when while around 2 Gy per fraction is the convention today, it has been showed that higher doses per fractionation might be better for abscopal effects, although too high doses might counteract the effect. Clearly, more research on the abscopal effect in general and also on how to induce this effect in a clinical setting is needed.

2.5 Spatially fractionated grid radiation therapy (GRID)

This chapter is based on the paper "A Current Review of Spatial Fractionation: Back to the Future?" (Billena and Khan 2019).

While conventional radiotherapy aims to give a homogenous dose to the target, GRID delivers the dose in a nonconfluent pattern, as illustrated in Figure 2.19. A normal GRID configuration is to have a sieve-like pattern, but common for all is to deliver the full dose to some areas, while other receive little or no dose. The pattern is made either by using a collimator block, multileaf collimators or a hybrid of the two. GRID is a strictly 2-dimensional method, but a 3-dimensional variation called lattice radiation therapy (LRT) has also been developed. Here, the dose is delivered in spheres inside the tumor volume. Spatially fractionated radiation therapy (SFRT) includes both these modalities.

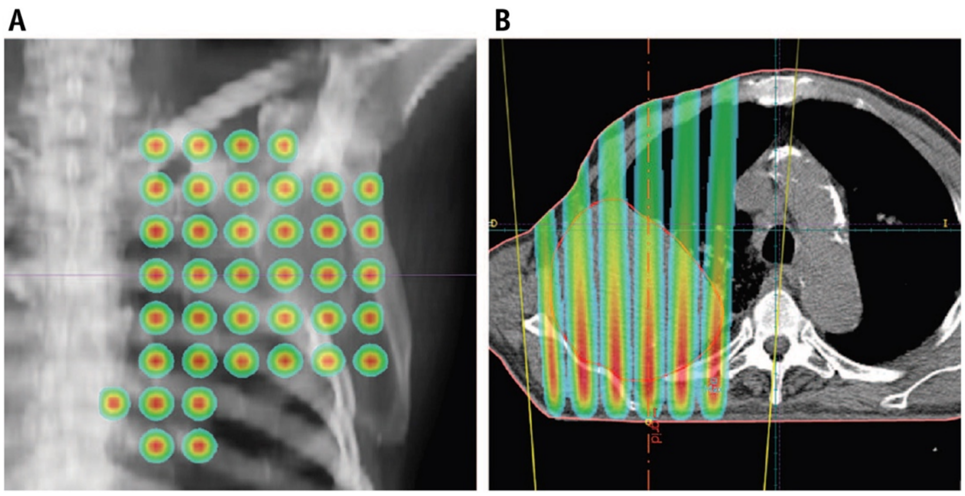


Figure 2.19: Example of a treatment plan using GRID pattern on lung cancer. Red indicates higher doses, while green is lower. ***A:*** *Anteposterior view.* ***B:*** *Transverse view. (Billena and Khan 2019)*

One obvious advantage of SFRT is that the volume receiving radiation is smaller, and therefore more normal tissue is potentially spared. This also enhances repair, since unirradiated cells can migrate and mediate repair in damaged areas. Higher doses can then be given, while still avoiding damage to normal tissue, especially the skin. With LRT the peak doses are also limited to be within the set tumor volume, which spares potential organs at risk.

In addition to volume sparing, SFRT has been shown to induce changes in the microenvironment of the tumor. This includes the bystander effect, with the bystander cells here being tumor cells in low dose regions. It was discovered that bystander tumor cells had less survival than expected for the low dose area. The low dose cells also had an overexpression of proteins associated with DNA repair, afpoptosis and cell cycle control. Some studies have found increases in TNFa (cytokine associated with tumor killing) and TRAIL (protein inducing apoptosis) when using SFRT. This increase is correlated with improved clinical response.

Another radiobiological effect of SFRT are microvascular alterations, especially the activation of acid sphingomyelinase (ASMase) in endothelial cells. ASMase is an important component for apoptosis induced by radiation. This changes the tumor microvasculature and hence also tumor growth. This has been demonstrated by knocking out ASMase, resulting in radiation not inducing apoptosis in those cells. In patients undergoing SFRT, ASMase activity was elevated corresponding with the good clinical response in those patients.

There has also been reports stating SFRT induces abscopal effects. Different studies of mice with lung carcinoma has shown that treating one tumor with SFRT reduces the growth of both. It was also shown that partial tumor irradiation was more effective than total irradiation, with more effect by irradiating two 10% volumes than one 20%. An increase in CD3+ T cells were found in both tumor sites, as well as up/down regulation of different cytokines, corresponding with the tumor growth inhibition. Some also suggest that GRID therapy therefore can be used as an immune activation in combination with conventional therapy.

Conventional therapy delivers dose in fractionations, one of the reasons being to overcome hypoxia and allow tumor cells to re-oxygenate. However, SFRT has shown to have greater tumor control than expected when using a high dose hypo-fractionated regimen. It is believed that the doses are high enough to overcome the tumor cells' hypoxic radioresistance. Also, the different radiobiological changes mentioned above, and the tumor killing deriving from it, may decrease the oxygen consumption and lead to reoxygenation. (Poh, Chua, and Wee 2018)

There have been done some clinical trials using GRID, and the outcomes have been promising with higher response rates than by conventional therapy. However, the studies have limitations, such as lack of control groups, heterogeneity in the study population, and inconsistent or uncontrolled combination with conventional therapy. Since GRID is a 2-dimensional delivery of dose, identical GRID configurations will deliver different doses depending on the anatomy. This makes the dose and delivery methods relative to target volume and organs at risk variable.

With new technology, new types of SFRT has been introduced. LRT has already been mentioned, but another way to restrict the dose is by using protons. This might be advantageous due to their way of depositing dose – low entrance dose, max dose at the Bragg peak and no exit dose. Clinical data using protons are however limited. Another type of SFRT is microbeam radiation therapy (MRT), using beams with a size of 25-50 microns. The spacing in between is typically 200-400 microns. MRT can give doses up to 100 Gy without noteworthy damage to normal tissue, but here too there are several challenges preventing the use of MRT clinically.

Although there are promising results regarding SFRT, it is clear that the benefits are not fully understood. This challenges a wider implementation of SFRT in a clinical setting, also because it represents a paradigm change from current clinical practice. More knowledge is needed on the cell response and radiobiological mechanisms in order for SFRT to be applied on a greater clinical scale.

3 Materials and methods

3.1 Dosimetry

The main purpose of this thesis is to establish an estimation of deposited dose when irradiating using GRID, in order to later elucidate the cell colony formation for the same irradiation conditions. This was to be done using both X-rays and protons, also for establishing any variation in the dose deposition from the two different radiation types. The strategy was to perform GRID irradiation and dose measurements with Gafchromic EBT3 films placed inside of cell culture flasks/petri dishes. As the EBT3 films darken when exposed to radiation it was possible to use these films to get a 2-dimensional map of the dose deposited, which then could be further analyzed and compared to cell survival assays. Monte Carlo simulations were also done to supplement the dose measurements.

3.1.1 X-ray experiments

The X-ray experiments were performed at the Roentgen radiation lab at the Chemistry building, UiO. A PANTAK PMC 1000 X-ray unit was utilized for the dosimetry and cell clonogenicity experiments, where the unit operated at 220 kV and 10 mA. At a source-to-detector-distance (SDD) of 60 cm, the X-rays were sequentially attenuated through a 0.70 mm Cu and a 1.52 mm Al filter.

GRID irradiation with X-rays was conducted using a specially designed Tungsten GRID. The GRID provides a striped irradiation pattern and is made with openings of 5 mm with 10 mm blocking in between. The outer dimensions of the GRID are 150 x 130 x 5 mm^2. It was made to perfectly fit and be attached to a polymethyl methacrylate (PMMA) cell flask holder. This flask holder was designed and built in-house to contain four T25 cell culture flasks (Nunclon, Denmark) for X-ray irradiation. The holder holds the flasks at a height of 1.5 cm. A figure of the experimental set up for GRID irradiation with X-rays can be seen in Figure 3.1.

In order to do GRID irradiation with Gafchromic EBT3 films, hereby referred to as films, the dosimetry of the X-ray chamber had to be established. It was necessary to know the dose rate and homogeneity at the Perspex plate (seen in Figure 3.1) – both for calibration of the films, and also to establish a placement of the PMMA cell flask holder where all four flasks receive approximately the same dose when irradiated. In addition, the dose rate inside of the T25 flasks when placed in the PMMA cell flask holder had to be determined, as this is where the cells are situated during irradiation.

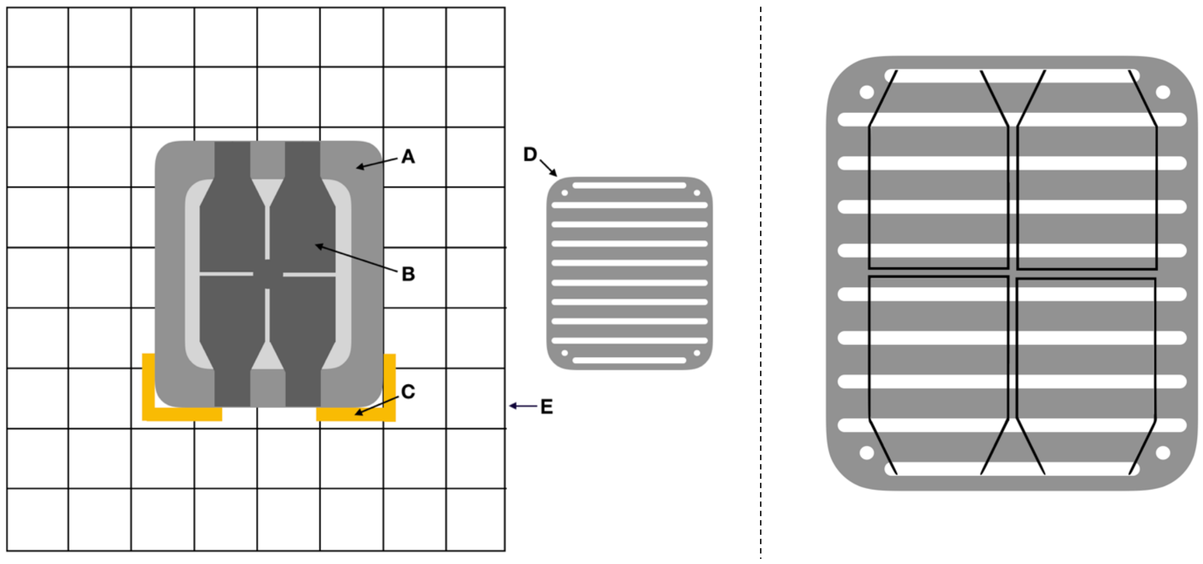


*Figure 3.1: **Left**: Experimental set up for GRID irradiation. **A:** PMMA cell flask holder. **B:** One of the four cell flasks. **C:** tape markings on plate. **D:** GRID collimator. Openings are 5 mm with 10 mm spacing between. When using the GRID, it is placed over the light grey area of the flask holder, covering the four cell flasks. **E:** Perspex plate placed at an SDD of 60 cm. **Right:** Sketch of GRID collimator with placement of the four T25 cell flask outlined. The four circular holes at the corners were for attachment.*

Point-by-point X-ray dosimetry on Perspex plate

The aim of this dosimetry was to allocate and outline the most homogenous exposure area at the Perspex plate, and also establish the dose rate in this area. The dose measurements were conducted with a KERMA calibrated IBA FC65-G ionization chamber (IBA Dosimetry, Germany) together with a Standard Imaging MAX-4000 electrometer (Standard Imaging, USA).

Dosimetric measurements were done directly on a Perspex plate positioned at SDD = 60 cm, where a flat two-dimensional coordinate system was drawn. An irradiation time of 1 minute was used. Initially, two measurements at two different spots near the centre of the plate was done to verify the stability of the X-ray unit and the reliability of the measurements. Sample measurements were then taken at selected spots across the plate, chosen based on previous knowledge on field homogeneity. The ion chamber was placed in the same direction for all measurements and taped to the plate to ensure stability and reproducibility in the dosimeter readings. An overview of the measurement readings is demonstrated in Figure 3.2. Subsequent interpolation was performed to estimate the missing grid points on the bench plate.

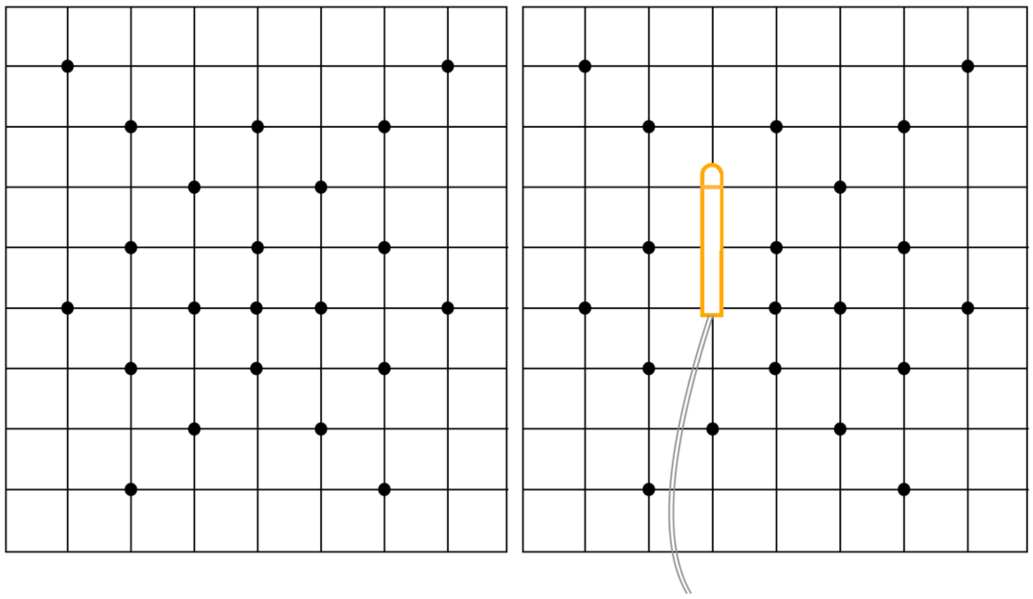


Figure 3.2: ***Right:*** *Perspex plate with marked spots where the ion chamber readings were done.* ***Left:*** *Example of how the dosimeter (in orange) was placed during measurements. The horizontal orange line indicates the center of the ion chamber.*

When starting the irradiation, the X-ray tube uses a few seconds to reach the set voltage. During these seconds, the established dose rate will not be constant, and for that reason a separate low-dose dosimetry was done. The ion chamber was placed in what was found to be the middle of the field, and irradiated for 5, 10, 15, and 20 seconds, with two measurements per timepoint. A linear fit to these points determined a low-dose rate calibration for doses up to 0.5 Gy.

Dose to water D is estimated as stated in the IAEA report TRS 277 (Andreo et al. 1987),

$$D = M_n N_K k_u \left(\frac{\overline{\mu}_{en}}{\rho}\right) P_u k_{\mathrm{TP}} \tag{3.1}$$

where M_n is the chamber reading, N_k is an air kerma calibration factor for the given beam quality, k_u corrects for the change in response as the spectral distribution changes when moving from air to water, $\left(\overline{\mu}_{en}/\rho\right)$ is the ratio of the mass energy absorption coefficient between water and air, averaged over the photon spectrum at reference depth, and p_u is the perturbation factor. These factors were found in the TRS 227 report. k_{TP} corrects for the change in temperature and pressure from the calibration to the measurement, and is given by:

$$k_{\mathrm{TP}} = \frac{273.15 + \mathrm{T}}{273.15 + \mathrm{T}_0} \frac{\mathrm{P}_0}{\mathrm{P}} \tag{3.2}$$

where T is the temperature in Celsius during the measurement, T_0 is = 20°C, P is the air pressure during the measurement and P_0 is 1013 Pa. An overview of the factors used in equation (3.1) is found in Table 3.1.

Table 3.1: Factors used in equation (3.1).

N_k [mGy/nC]	k_u	$(\bar{\mu}_{en}/\rho)$	p_u	k_{TP}	k_{TP} (low-dose dosimetry)
43.77 ± 0.39	1	1.075	1.02	1.010	1.018

The uncertainty of equation (3.1) is found using the expression for error propagation:

$$\sigma_y^2 = \sum_i \left(\frac{\partial y}{\partial x_i}\right)^2 \cdot \sigma_{x_i}^2 \tag{3.3}$$

The only uncertainties in equation (3.1) lies in N_k and the chamber readings M_n, giving the full expression,

$$\sigma_D = \sqrt{\left(\frac{\partial D}{\partial N_k}\right)^2 \sigma_{N_k}^2 + \left(\frac{\partial D}{\partial M_n}\right)^2 \sigma_{M_n}^2} \tag{3.4}$$

X-ray dosimetry within cell culture flasks

In this dosimetry procedure the aim was to establish dose distribution and value inside T25 flasks with the setup in seen in Figure 3.1. The position of the PMMA cell flask holder as shown in this figure was determined and marked based on the homogeneity measurements done at the Perspex plate. Two consecutive measurements of one minute each was performed at each flask position with the ion chamber inserted into a T25 flask, as shown in Figure 3.3. The dose rate at each flask position as well as the average of all four positions, was calculated using equation (3.1) with factors from Table 3.1, and the belonging uncertainties was calculated from equation

(3.4). A separate low-dose dosimetry was carried out using the same time points as at the Perspex plate, with two measurements for each time point at each of the four positions, still with the ion chamber inside a T25 flask.

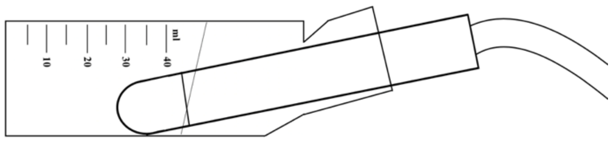


Figure 3.3: Sketch of the ionization chamber inserted into a T25 cell culture flask.

Calibration of Gafchromic EBT3 films

Gafchromic EBT3 films darken when irradiated, and the net optical density (netOD) can be correlated to a dose. We therefore want to use these films to get a 2-dimensional map of the dose delivered in the T25 flasks when irradiating with GRID. To do this, a calibration of the films was needed, and a suitable model for converting the netOD into dose had to be established.

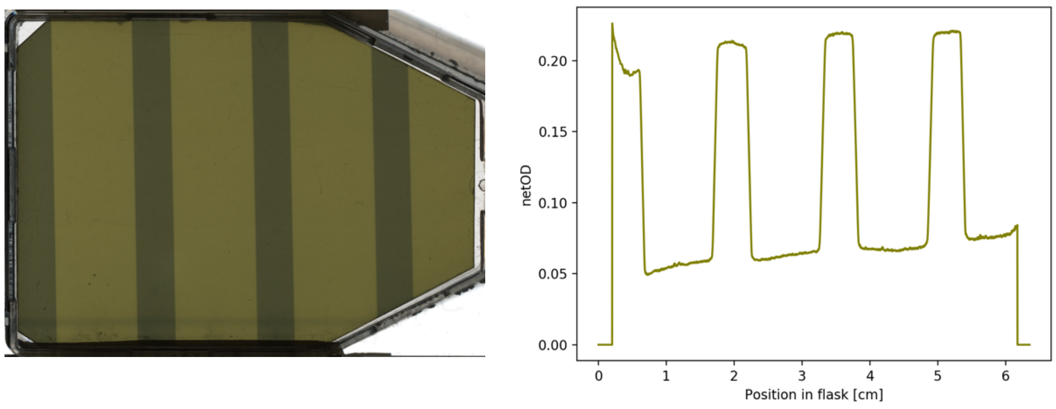


Figure 3.4: Example of a film irradiated with GRID, and the belonging netOD profile taken through the film. We want to trannsfrom the netOD to dose, to establish the dose delivered at various positions of the film.

Gafchromic EBT3 films (Lot # 11031501) were used, consisting of an active layer of 28 µm sandwiched between a polyester base of 125 µm, Figure 3.5 shows the structure while the atomic composition is found in Table 3.2. The dose range is between 0.1 Gy - 20 Gy with an optimum range from 0.2 Gy - 10 Gy, with a minimal response difference for energies of 100 keV into the MeV area (GAFChromic™). The entire protocol used regarding the films is based on the recommended protocol from the vendors (GAFChromic™ 2016). The films were always handled using gloves to avoid finger marks and scratches and kept in the dark as much as possible to avoid any potential response in the films due to UV light. As the films carry a scanning orientation dependent property extra care was taken to make sure they were cut in the same direction, where a corner was marked to keep track of this. The films used for calibration were cut into squares of 5.9 x 6.3 cm^2, in all 16 films.

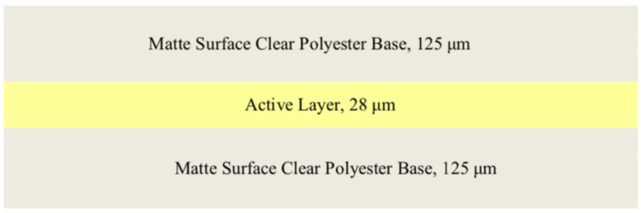

Matte Surface Clear Polyester Base, 125 µm
Active Layer, 28 µm
Matte Surface Clear Polyester Base, 125 µm

Figure 3.5: Structure of Gafchromic EBT-3 films. (GAFChromic™)

Table 3.2: Atomic composition of EBT3 film, expressed as % by weight (Sipilä et al. 2016).

	Density (g/cm^3)	Thickness (cm)	H	Li	C	O	Al
Polyester base	1.35	0.0125	4.2	-	62.5	33.3	-
Active layer	1.20	0.0028	8.8	0.6	51.1	32.8	6.7
Polyester base	1.35	0.0125	4.2	-	62.5	33.3	-

A schematic display of the set-up is shown in Figure 3.6. Due to the protective cap of the ionization chamber, the surface wall of the sensitive material was elevated 7.5 mm above the Perspex plate during the X-ray dosimetry there. Consequently, a Nylon6 slab of 7.5 mm thickness was placed underneath the films to complement the height of the sensitive volume of the dosimeter. Furthermore, a Nylon6 slab with 2 mm thickness was placed over the films to ensure electronic build-up at the active layer. Two films were irradiated per dose point, with the following doses: 0.1 Gy, 0.2 Gy, 0.5 Gy, 1 Gy, 2 Gy, 5 Gy and 10 Gy, in which two additional unirradiated films served as controls. The experiment was conducted twice.

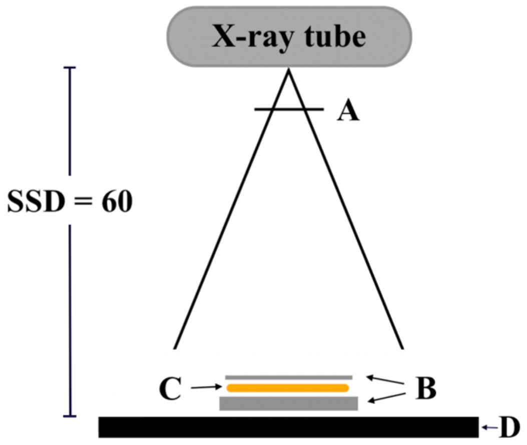


Figure 3.6 Experimental setup of film dosimetry. ***A:*** *filtration of x-rays.* ***B:*** *Nylon6 with a thickness of 7.5 mm underneath the film and a thickness of 2 mm on top of the film.* ***C:*** *Gafchromic EBT-3 film.* ***D:*** *Perspex plate at an SDD of 60 cm.*

In order to correlate the optical density to a dose, a calibration model was needed. Three candidate models are presented in Table 3.3, where each model was applied to all three color channels (red, green, blue) as well as the weighted mean (greyscale) image. Model 1 is a widely used fit and thus a lot of information concerning this model can be found in the literature. Model 2 is the vendors own recommendation to use as a fit to Gafchromic EBT3 film, and model 3 was tested because a former master student previously had found it to be the best fit. Model 4 however, was added later on, after preliminary testing of the models showed unsatisfactory uncertainties in the fitting. It is a modifies version of Model 1, made after an inspection of the statistical significance of the parameters in Model 1, removing one term to make it univariate when *n* is fixated, as is explained further down.

Table 3.3 Models tested for calibration of Gafchromic EBT3 films.

Model 1 (Devic et al. 2004)	$a \cdot \mathrm{netOD} + b \cdot \mathrm{netOD}^n$
Model 2 (GAFChromic™)	$a + \dfrac{b}{\mathrm{netOD} - c}$
Model 3 (Castriconi et al. 2017)	$a \cdot \mathrm{netOD} \cdot \mathrm{e}^{\,b \cdot \mathrm{netOD}}$
Model 4	$a \cdot \mathrm{netOD}^n$

In the models shown in Table 3.3, *a*, *b*, *c* and *n* are fitting parameters while netOD is the measured net optical density given by equation (3.5) (Devic et al. 2004),

$$\text{netOD} = \log_{10}\left(\frac{I_{unexp} - I_{bckg}}{I_{exp} - I_{bckg}}\right) \tag{3.5}$$

With the uncertainty found from the error propagation (equation (3.3)):

$$\sigma_{\text{netOD}} = \frac{1}{\ln 10}\sqrt{\frac{\sigma^2_{I_{unexp}} + \sigma^2_{I_{bckg}}}{\left(I_{unexp} - I_{bckg}\right)^2} + \frac{\sigma^2_{I_{exp}} + \sigma^2_{I_{bckg}}}{\left(I_{exp} - I_{bckg}\right)^2}} \tag{3.6}$$

where I_{exp} is the measured intensity of an irradiated film, I_{unexp} is from the unirradiated control films, given as:

$$I_{unexp} = \frac{\sum_i \left(I_{unexp_i}/\sigma_i^2\right)}{\sum_i\left(1/\sigma_i^2\right)} \tag{3.7}$$

With the corresponding uncertainty:

$$\sigma^2_{I_{unexp}} = \frac{1}{\sum_i\left(1/\sigma_i^2\right)} \tag{3.8}$$

where the summation is done over all unirradiated films. I_{bckg} is the intensity from an opaque film with no transparency, but as we used two opaque films, equation (3.7) and (3.8) were used to find I_{bckg} and $\sigma^2_{\mathrm{I}_{bckg}}$ as well.

Treating the power n in Model 1 and Model 4 as a fitting parameter introduces a higher fit uncertainty while only negligibly improving of the sum of residuals (Devic et al. 2004). An iterative search for the optimal n was therefore first done by varying it from 0 to 5 with steps of 0.25. Fitting the data to the models was done using python, with a built-in function using the Levenberg-Marquardt algorithm. This could be applied on all models as it is capable of handling both linear and non-linear regression. The Levenberg-Marquardt algorithm is further elaborated on in section 3.6.1.

To evaluate the goodness of the different fits, the Corrected Akaike Information Criterion (AIC_C) was used. This is specially designed for small sample sizes and uses the residual sum of squares (RSS) as an estimator whilst penalizing overfitting of both linear and non-linear models. More details on AIC_C in section 3.6.2. However, AIC_C solely scores the relative quality between candidate models, and is therefore not an absolute quality measure. Hence, in order to acquire an absolute estimate of goodness of fit, R^2 values was found for the linear models. This gives an indication of the quality of all models, as AIC_C then gives the relative ranking between the different fits. The RSS values was also obtained for all models. An analysis of the models' residuals was done through locally estimated scatterplot smoothing curves (loess curves), and in addition, the total uncertainty per dose was evaluated. The latter was done using the error propagation from equation (3.3), with $y = D_{tot}$, and σ_{x_i} (i = 1,2,3,4) as the standard deviation of the fitting parameters *a*, *b*, and *c*, and the netOD. This way, both the experimental uncertainty and the uncertainty relating to the fit was accounted for, and by separating the uncertainty equations into the terms related to the uncertainty of the measured netOD (σ_{netOD}), the experimental uncertainty can be found, and similarly the uncertainty originating from the fit itself is found by separating into terms related to the parameter uncertainty (σ_a, σ_b, σ_c). The uncertainty is of course discrete and dependent on each film used for calibration, as it is a function of both netOD and σ_{netOD}. However, when using a model to find the dose related to a particular netOD, we wish to know the uncertainty of this dose estimate. This was done by finding the average netOD and σ_{netOD} of the films given the same doses, where the average σ_{netOD} was found through standard deviation of the mean, $\sigma_{mean} = \sigma/\sqrt{n}$, *n* being the number of films given the same dose. A spline interpolation was consequently done with a high sampling rate as this enables a good estimate of the uncertainty for any dose in between the lowest and highest dose used for calibration. The full equations for the uncertainties of the different models are listed in Appendix A.

It has been shown that the size and location of the Region of Interest (ROI) affects the resulting netOD, and thus the outcome of the ensuing EBT3 film calibration, (Gholizadeh Sendani et al. 2018). Sendani et al. recommend a 4 x 4 mm^2 ROI when the field size is between 10x10 mm^2 and 100x100 mm^2. Based on this work, and also looking at ROIs chosen by other studies using Gafchromic EBT3 films (Aldelaijan and Devic 2018; Devic et al. 2004; Krzempek et al. 2018; Castriconi et al. 2017), we selected a ROI of 4x4 mm^2 placed at the centre of the film, shown in Figure 3.7. For validation purposes, the calibration was also performed with a 3x3 mm^2 ROI as a comparison. The same goodness of fit assessment was carried out using both ROIs, where the model selection was applied to one randomly chosen calibration film for testing.

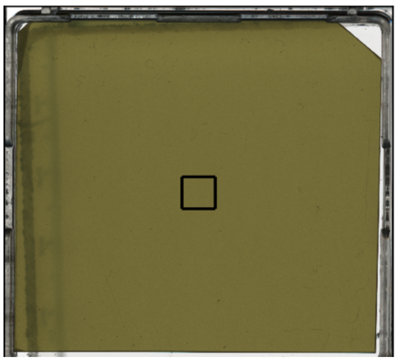


Figure 3.7: Example of a calibration film with ROI of 4x4 mm^2 placed at the center marked in black.

X-ray film dosimetry in GRID configuration

Finally, the 2-dimensional dosimetry in the T25 flask when irradiated with GRID could be done. Prior to irradiation the films were cut to fit into an empty T25 flask. A slit was cut into the back of one of the T25 flask where the films could be inserted and removed, and a small piece of tape was used to do this smoothly. A side view of the setup is shown in Figure 3.8, with the placement of the PMMA flask holder identical to that in Figure 3.1. The Nylon6 slab of 2 mm thickness was placed on top of the GRID to get electronic build-up at the active layer of the films.

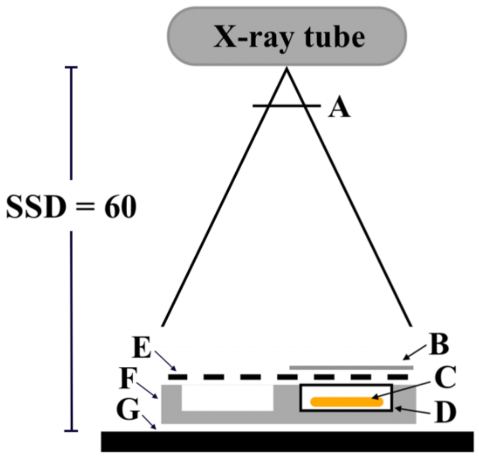


Figure 3.8: Experimental setup of film dosimetry with GRID setup. ***A:*** *X-ray filtration.* ***B:*** *Nylon6 with 2 mm thickness.* ***C:*** *Gafchromic EBT-3 film.* ***D:*** *T25 flask.* ***E:*** *GRID collimator.* ***F:*** *PMMA flask holder, keeping the T25 flasks 1.5 cm above the Perspex plate.* ***G:*** *Perspex plate.*

The films were irradiated while inserted into a T25 flask, with two films irradiated in each of the four positions in the PMMA flask holder – in all eight films with GRID, as well as two films in one position without the GRID to serve as control. All films were given 5 Gy. The experiment was conducted twice.

3.1.2 Proton experiments

A proton beam deposits dose differently than an X-ray beam, suggesting the dose deposition when using GRID with protons might differ from GRID with X-rays. As protons are up and coming in a clinical setting, we wish to explore using GRID also with protons to see how the dose is deposited in comparison to the X-ray GRID experiments. Thus, we both want to compare the dose distributions in themselves, but also the dose delivered vs cell survival for the two different radiation types.

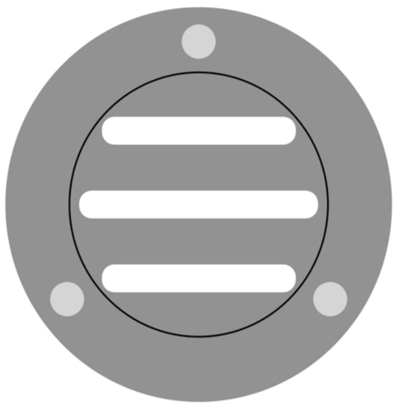


Figure 3.9: Specially designed Tungsten GRID for proton irradiation. Openings of 5 mm with 10 mm shielding in between. The inner dark circle outlines the position of a 60x15 mm petri dish. The three holes around the edge are for attaching the GRID to a special holder as the irradiations were done with the GRID placed vertically.

The proton experiments were carried out at the Oslo Cyclotron Laboratory (OCL). The experimental set-up is shown in Figure 3.10. A Tungsten GRID was specially designed and build in-house with the same properties as the Tungsten GRID for X-rays: Openings of 5 mm with 10 mm blocking in between. An illustration of the GRID is show in Figure 3.9. This GRID was made to cover a single 60x15 mm petri dish (Nunclon, Denmark). The proton beam is horizontal, and everything had therefore to be irradiated in a vertical position. Consequently, a special holder was made to hold the petri dish and GRID vertically.

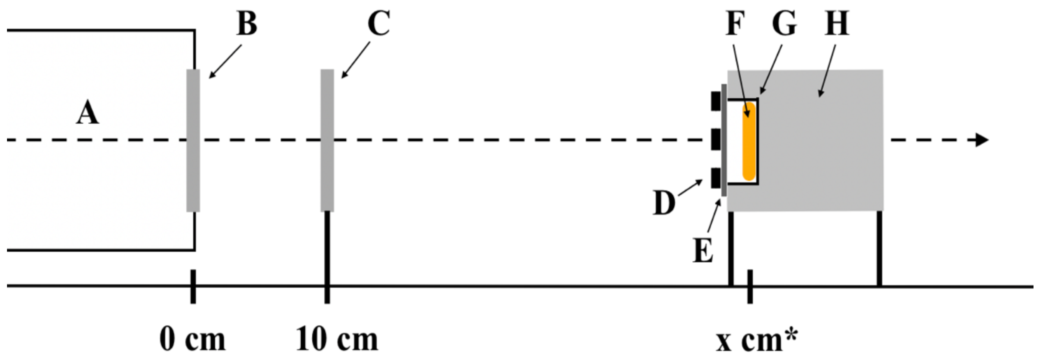


Figure 3.10: Setup for GRID irradiation with protons. ***A:*** *Proton beam.* ***B:*** *Beam exit window with scattering foil.* ***C:*** *Transmission chamber.* ***D:*** *Tungsten GRID.* ***E:*** *0.130 mm parafilm.* ***F:*** *Gafchromic EBT3 film.* ***G:*** *Petri dish.* ***H:*** *holder for petri dish and GRID. *distance from the beam exit window to the film was different from day to day and had to be established before each experiment.*

A nominal proton beam energy of 15.5 MeV with a 1 nA current was generated by a MC-35 cyclotron (Scanditronix, Lund, Sweden). Directed through the beam guides under vacuum, the protons were then spread at the beam exit window through a 52 µm tungsten scattering foil. A Monitor Chamber type 7862 (PTW) transmission chamber was placed at a 10 cm distance to the beam exit window, monitoring the beam intensity. A UNIDOSE E (PTW) Electrometer was used together with the PTW Monitor Chamber. Dose measurements were done with a dose to water calibrated PTW Advanced Markus Electron chamber (PTW-Freiburg, Germany) together with a MAX-4000 electrometer (Standard Imaging, USA). A special holder was made for the Markus chamber, with a slot of 60 mm x 65 mm in front of the chamber where solid water could be placed. All instruments used – including the petri dish- and GRID holder – were attached to a horizontal steel rail to ensure a rigid setup, as well as centered according to the midpoint of the tungsten scattering foil using a cylindrical laser holder with a coaxial laser beam. The beam centration was assed visually using an EBT3 film at the beam exit window when setting up the equipment.

The experiments were conducted in front of the Bragg peak, see Figure 3.11, were they receive around 20% of the maximum dose of the proton beam. As the proton beam is known to fluctuate, this positioning was chosen because it gives a stable dose even with these possible fluctuations of the beam. Dosimetric measurements were done before each experiment, using the Markus chamber and the transmission chamber. This was done to localize the Bragg peak, and to establish at what distance from the beam exit window to place the gafchromic films in order to get to the desired location of the Bragg peak. The dosimetric measurements also determined what monitor units were required to deliver the desired doses. The average of three consecutive measurements of the monitor chamber (MC) and ionization chamber (IC) at the desired depth gave the factor, equation (3.9), used to calculate the monitor units (MU), equation (3.10). The setup for dosimetric measurements is shown in Figure 3.12.

$$\text{Factor} = \frac{\text{IC [nC]}}{\text{MC [}\mu\text{C]}} \tag{3.9}$$

$$\text{MU[}\mu\text{C]} = \text{D[Gy]} \cdot \frac{1}{1.411\ \text{[Gy/nC]}} \cdot \frac{1}{\text{Factor}} \tag{3.10}$$

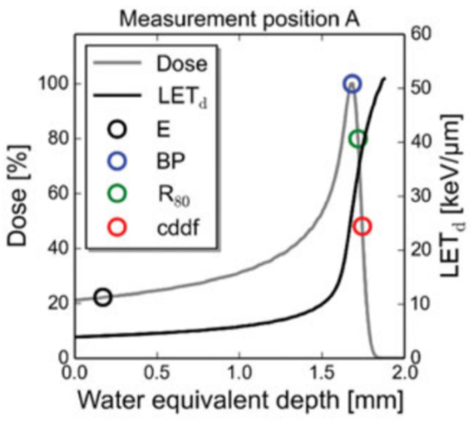


Figure 3.11: Visualization of a Bragg peak. The position marked in black, E (Entry), represent where in the Bragg peak our experiments were performed. (Dahle et al. 2017)

The EBT3 films were cut into circles with a radius of around 26 mm, made to exactly fit into the petri dishes they were irradiated inside of. The petri dish was covered with a 0.130 mm thin Parafilm M Laboratory Film (Pechiney Plastic Packaging, Menasha, WI 54952) instead of the petri dish lid, which together with the distance from the exit beam window to the films was done to get to the desired location at the Bragg peak. The films were from the same batch as the films used with X-rays, and all the same precautions were taken. As the film used for proton irradiation were cut to circles, a small pen mark was drawn onto them to keep track of orientation.

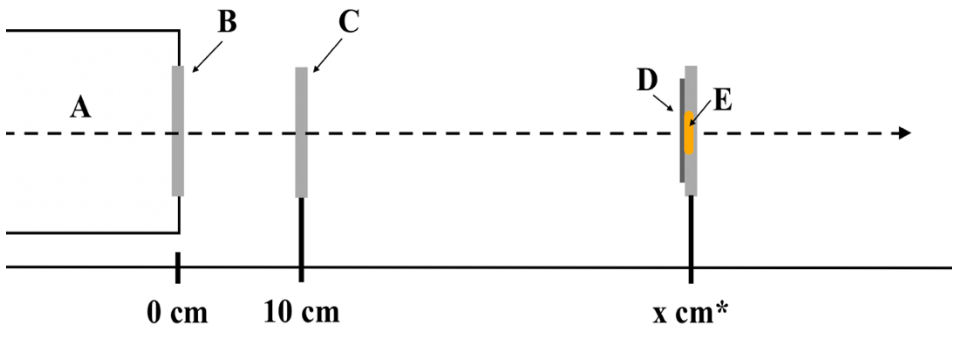


Figure 3.12: Setup for proton dosimetry. ***A:*** *Proton beam.* ***B:*** *Beam exit window with scattering foil.* ***C:*** *Transmission chamber.* ***D:*** *0.130 mm parafilm.* ***E:*** *Markus chamber. *various distances from the beam exit window to the dosimeter was tested in order to find the desired location in the Bragg peak.*

In order to compare the dose distributions with GRID for protons and X-rays, it was important to know whether the potential differences actually came from the different radiation types reacting differently to the GRID. To rule out other factors, and also to verify the calibration done with X-rays, EBT3 films were first irradiated with protons in an open field. These could then be compared to the calibration series done with X-rays, and the possible deviation taken into consideration when later analyzing the different GRID dosimetries.

First, films were irradiated in an open field, using doses of 2, 5 and 10 Gy with two films per dose. Then two films were irradiated with GRID, both receiving 5 Gy. Dose is delivered by starting and stopping the beam manually, monitoring the transmission chamber though a camera and stopping when the required MU is reached. As the fluence of the proton beam vary it can be quite challenging to hit the targeted dose exactly, and the actual given MU were therefore noted for each irradiation. All films were irradiated in the same orientation using the pen mark, and the GRID was always placed with the stripes horizontally. The experiment was done twice.

3.2 Monte Carlo Simulations

Monte Carlo simulations are viewed as the golden standard for dose calculations, and it was therefore desirable to do Monte Carlo simulations with the GRID setup as a verification of the dose distribution found through the Gafchromic EBT3 film dosimetry.

Monte Carlo simulation (MC) is an approach to problem solving and modelling of complex processes by using algorithms to perform random sampling. The concept is to do repeated simulations using random numbers and probability distributions in order to approximate a solution. The problems or models often depend on stochastic variables, although it can also be used to solve deterministic problems by introducing randomness. Common usages of MC are: Sampling, where a stochastic prosses is run through and observed several times, often mimicking a real-life random system. Estimation, where numerical quantities related to a model are found, like the estimation of multidimensional integrals. Optimization, where randomness is used to efficiently optimize complex models. MC is often used when other approaches are difficult or even impossible. (Kroese. et al. 2014)

3.2.1 FLUKA Monte Carlo

FLUKA is a Monte Carlo code specially designed to calculate particle transport and interaction with matter. It can simulate around 60 different particles with high accuracy, including photons and electrons with energies from 1 keV to thousands of TeV, and handle complex geometries. FLUKA operates with microscopic models and ensures consistency and the enforcements of conservation laws, as well as checking the results against experimental data at each step, using predictivity if no experimental data is available. This allows final predications to be made with a minimal of free parameters, and complex results, properties and scaling laws arises naturally from the underlying physical models. (Ferrari et al. 2005)

Simulated setup of the irradiation experiment

Radiation transport was simulated using the FLUKA Monte Carlo (MC) code version 2011-3.0 0 (Fasso. et al. 2003; Ferrari et al. 2005; Böhlen et al. 2014; Battistoni et al. 2016) conducted by Delmon Arous, PhD-student. The purpose was to achieve a theoretical basis of the EBT3 dosimetry measurements that could be directly related to the experimental measurements from the scanned EBT3 films. In doing so, the simulated setup of the X-ray experiment was accurately implemented taking into account meticulous distances and component thicknesses, elemental compositions and mass densities (see section 3.1.1 (Figure 3.8)) for overview of the irradiation scheme), including modeling of the EBT3 films within the T25 flasks. The geometrical scheme of the experimental irradiation system was defined and outfitted through flair (FLUKA Advanced Interface) version 3.0-12, where the geometry editor in flair, Geoviewer version 3.0-12, was also applied to visualize and affirm the irradiation layout of the simulations.

Beam configuration and physics settings

The polyenergetic photon beam energy was set to 220 kV, with a flat beam divergence of 290 mrad in order to uniformly cover all four modelled T25 flasks at a distance of 60 cm SDD. The X-ray spectra was simulated for 2×10^7 photons, during the course of a single simulation cycle, which were emitted through the Cu + Al window of the X-ray tube, where 5 independent cycles were carried out. In total, the number of particles forming the irradiation field was 10^8 primary photons. Furthermore, the MC acquisition was performed by adopting default FLUKA physics settings (cf. "PRECISIOn" defaults in the FLUKA manual). Here, the amount of kinetic energy loss is was at 2 % per step size. Also, to limit ionization fluctuations, transport and production thresholds were implemented. Both for photons and electrons, the transport, i.e., below which energy is deposited uniformly and locally, and production cutoffs were set to 1 keV.

Scoring

Algorithms for scoring energy deposition in 3-D cylindrical meshes were applied by the use of the "USRBIN" card in FLUKA. A two-dimensional dose distribution was scored in a spatial regular mesh encompassing the volume of the constructed EBT3 film region of $6.48 \times 4.26 \times 0.0278$ cm^3 in which was split into $400 \times 400 \times 1$ uniformly sized bins, respectively. Consequently, this rendered a 0.1616×0.1062 cm/pixel lateral spatial resolution of the modeled films. The two-dimensional lateral scoring area was chosen to fit the entire EBT3 film area as to have a two-dimensional matrix that can serve to assess the EBT3 images by pixel.

3.3 Cell experiments

The cell experiments were conducted by Magnus Børsting, MSc-student.

3.3.1 Cell line and seeding

The cell line A549 comes from a lung cancer tumor resembling type II alveolar epithelial cells. The cell line originates from a study by Giard et al., (Giard et al. 1973). They were grown in a BioWhittaker 1:1 mix of Dulbecco's Modified Eagle Medium (DMEM) and F12 with 15 mM Hepes and L-Glutamine, with a 10 % of fetal bovine serum added. This medium must be replaced twice a week, and the cells must be diluted to ensure exponential growth and a sufficient amount of nutrients. To enable this, BioWhittaker Trypsin EDTA mix with 200 mg/L Trypsin and 170.000 U of Versene (EDTA) per liter of trypsin was used, before transferring the cells into new flasks. An incubator (Thermo, Usbekistan) in the cell lab on the third floor of the Chemistry building, UiO, served as a storage container for the cell flasks, providing an environment with 5 % CO_2 concentration and a temperature of 37°C. This mimic the environment of a human body, making the cells move through the cell cycle and replicate.

T25 flasks were seeded with 30 000 cells each. The seeding started with moving the medium and trypsin into a Grant JB Aqua 18 water bath set to 37-degree Celsius, where it sat for 15 minutes. The cells were always handled in a LAF bench (SAFE 2020; Thermo Scientific, USA) wiped down with 5 % virkon (DuPont, UK) and sprayed with MQ water. After preparing and thoroughly cleaning the equipment with 70 % ethanol, the cell flask was collected from the incubator and disinfected with ethanol, before moving the medium washing with 1.5 ml trypsin and then adding 3 ml trypsin. After 5 minutes incubation time the cells were drawn up through a disposable needle (Misawa Medical Industry, Japan) into a 2 ml syringe (Becton Dickinson, Spain), and sprayed into the bottom of the flask. This was repeated until the cells were seen through a microscope well separated. 3 ml of medium was added, before the cells were moved into a 15 ml tube (Sarstedt, Germany) and centrifuged for five minutes. After centrifugation, the supernatant was removed, and the crystallized cells resuspended in 6 ml of fresh medium. The cell density was then found by resuspending the cells with a pipette (Sarstedt, Germany) and taking around 0.1 ml from the middle of the tube into a Bürker chamber. Two chambers were filled, and each of the nine squares in the chambers were counted using a microscope. Excluding the highest and lowest outlier, the average of the remaining squares was used, and the cell solution diluted to a density of 30 000 cells/ml. 1 ml of this diluted mixture was added to each T25 flask together with 4 ml pure medium. This gave a density of 30 000 cells per flask, with a total of 5 ml medium. The cell dished used for proton irradiation was seeded identically.

3.3.2 Irradiation

X-rays

Irradiation of the cells were done with the Pantak PMC 1000 X-ray unit, UiO. The setup was as shown in Figure 3.1, analogous to the X-ray irradiation of Gafchromic EBT3 films. The cells were kept in a heated room next to the X-ray unit before irradiating to keep any temperature loss at a minimum. Also, the GRID collimator and PMMA cell flask holder was stored in a warmed-up lab, preventing the cells from any contact with cold materials. The cells were irradiated using 2 Gy, 5 Gy and 10 Gy, four replicates per done, one in each of the four positions in the PMMA cell flask holder. This was done both with the GRID and also in an open field to use for comparison. In addition, four cell flasks were kept unirradiated to serve as controls. The experiment was conducted twice.

Protons

The proton irradiations were done at the Oslo Cyclotron Laboratory (OCL), and a LAF bench (NuAire, China) next to the cyclotron served as a cell lab to handle the cells. The setup is as shown in Figure 3.10, analogous to the proton irradiation of Gafchromic EBT3 films. As the cell medium would shield the cells from the proton beam, it was poured out right before irradiation, removing any lasting drops with a cotton swab. The cells were irradiated using 2 Gy, 5 Gy, 10 Gy and 13 Gy, with three replicates each dose. All irradiation was done using the GRID. Four cell dishes were kept unirradiated to serve as control. The experiment was conducted twice.

3.3.3 Fixating and staining

The cells were given six days to replicate after irradiation, before they were fixated (killed). This was done by done by first removing the medium and washing the cells with phosphate-buffered saline (PBS), removing any loose biological residuals from the medium or cells. Then 3 ml of ethanol was used as a fixator as it is highly toxic to the cells. It was added to the cell flasks and left on for five minutes before removing. The cells were then stained by adding 3 ml of methylene blue for five to ten minutes, before removing it with a pipette and washing residue of with tap water. The staining was done to facilitate later counting of cell colonies, since they barely are visible in a natural state.

3.4 Scanning

The main purpose of the EBT-3 dosimetry assessment was to establish an estimation of deposited dose that can elucidate the survival of colony formation assays undergoing the same irradiation conditions. Thus, a one to one comparison of the dosimetry films and the cell assays must be possible. All films and cells assessed were scanned utilizing a flatbed laser scanner

(Epson Perfection V850 Pro), providing RGB images with a resolution of 1200 dots per inch (dpi) and 48-bit depth. No prior filtering nor adjustments were performed on the captured images during scanning procedure with the scanner software (EPSON Scan v3.9.3.3). A protocol was developed to stabilize the lamp to produce liable digital images: The scanner was on for 30 minutes prior to the scans, and 10 warm up scans were conducted.

3.4.1 Film scans

The films used for calibration were cut down into squares of approximately 43 x 40 mm^2, small enough to fit into a T25 flask. One T25 flask was cut in two, and the films were scanned inside with the top half loosely placed on top. This included the films receiving X-ray GRID irradiation. A template was used to ensure a reproducible positioning, and all X-ray films were scanned at the same position. Original scans were performed 25-30 hours after irradiation, however, due to unforeseen events they could not be used, and new scans were performed around 1 and 2 months after the respective irradiations.

The films used for proton irradiation, however, was scanned using a different protocol, and due to unforeseen events (corona pandemic), they could not be re-scanned to match the protocol used on the X-ray films. The scans used were therefore with a resolution of 150 dpi, scanned directly on the scanner bed with a glass plate placed over, adhering the films to the scanner. These scans were performed 25-30 hours after irradiation.

3.4.2 Cell scans

T25 flasks from X-ray experiments were scanned using the same template as the X-ray films, enabling a one to one comparison of the dose deposition and cell survival. The petri dishes from proton experiments were also scanned using the same protocol and template as the T25 flasks. This was necessary for further analyses of the cell colonies, even though it meant a one to one comparison to the proton films not would be possible without any preprocessing.

3.5 Image processing

Having obtained scans of all cell assays and irradiated film, some image processing was necessary in order to do various comparisons. This was particularly true for the cell assays, as we wanted a way to quantify not only the number of colonies in a cell flask, but also the positioning of them as this is vital information with respect to the GRID experiments. A possible way which was experimented with was to get the share of colored pixels (indicating a colony is there) vs white pixels (indication no colony) per pixel row. To do this, the images was first processed using morphological filtering and a global thresholding (see below). However, it soon became clear that more advanced methods for cell colony detection was necessary, and this was dealt with by PhD. Student Arous, D.

3.5.1 Morphological filtering

The cell flasks do not only contain viable colonies, dead cells are also stained and visible from the scans. These dead cells are clearly not to be considered when finding the surviving fraction of the cell flasks, and can be considered background noise which should be removed. One way of doing this is through morphological filtering. Morphological operations were also used in the preprocessing steps of the final cell colony counter algorithm.

Morphological operations are used as a step in preprocessing and analysis of images, by modifying the shape of objects using local operations and structure elements. The basic operations are erosion, dilation, opening and closing, and can help to remove noise, link objects together or separate them, or find contours or certain patterns in an image to mention some. The operations are easy and relatively fast – of course depending on the size of the images.

An erosion, shown in Figure 3.13, applied to a set of foreground pixels *A* using a structure element *B* defined as

$$A \ominus B = \{ z \mid (B)_z \subseteq A \} \tag{3.11}$$

where *z* are foreground values. Erosion shrinks objects, also from the inside if it contains holes, removes protrusions and expands indentations. The degree of erosion depends on the size and shape of the structure element.

A dilation, shown in Figure 3.13, of *A* by *B* is defined as:

$$A \oplus B = \{ z \mid (\hat{B})_z \cap A \neq \emptyset \} \tag{3.12}$$

Dilation expands objects, fills holes and smooths indentations. The degree of dilation is dependent on the size and shape of the structure element. Dilation and erosion are duals with respect to complementation and reflection. If the structure element is symmetric, the dilation of A is the same as an erosion of its background followed by complementing the result.

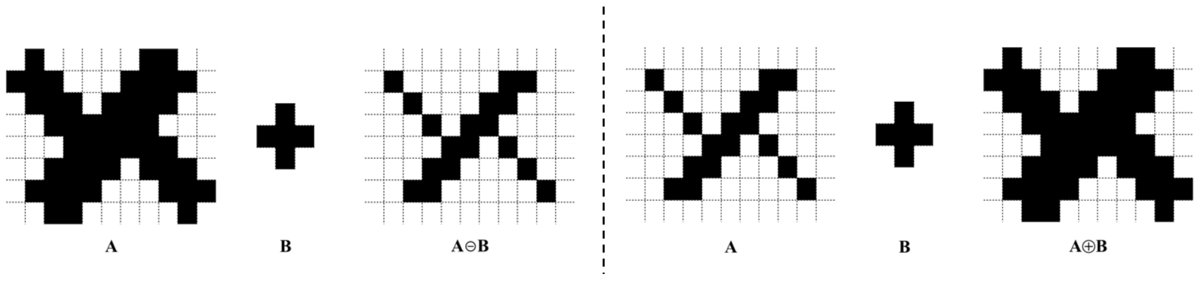


Figure 3.13: Examples of erosion (left) and dilation (right) of a figure A with a structure element B.

A morphological opening is an erosion followed by a dilation, using the same structuring element for both operations,

$$A \circ B = (A \ominus B) \oplus B \tag{3.13}$$

This will "open up" structures without shrinking them, compared to erosion only. Contrary to this, a morphological closing, Figure 3.14, is a dilation followed by an erosion,

$$A \cdot B = (A \oplus B) \ominus B \tag{3.14}$$

which will close small openings without expanding the object as a dilation alone would have done. Opening and closing are also duals, similar to erosion and dilation.

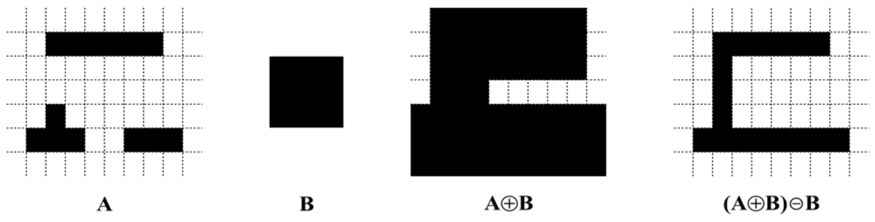


Figure 3.14: Example of a morphological closing of figure A using structure element B.

3.5.2 Optimum global thresholding using Otsu's method

Even with the removal of single cells through filtering, not all cell-groups left are actually to be considered colonies, as a colony is defined as a group of 50 cells or more. A global thresholding was therefor done.

Thresholding is the separation of the pixels in an image into a foreground and a background class. Viewing this as a statistical-decision theory problem, with the goal of minimizing the average error incurred, gives the solution known as Bayes decision function. However, this involves highly complex calculations even when simplified, so a good alternative is to use Otsu's method. This is a histogram-based operation, which also approaches the problem statistically by trying to maximize the between-class variance. The principal idea is to yield the optimal separation between the mentioned classes in terms of their intensity values.

If $P_1(k)$ is the cumulative sum of pixel values up to intensity level k, m(k) is the cumulative mean up to intensity level k, and m_g is the global intensity mean, the between class variance is given as:

$$\sigma_B^2(k) = \frac{[m_g P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]} \tag{3.15}$$

The optimum threshold value k^* is then:

$$\sigma_B^2(k^*) = \max_{0 < k \leq L-1} \sigma_B^2(k) \tag{3.16}$$

where L is the number of intensity values. The threshold is thus obtained by evaluating equation (3.15) for all integer values of k and selecting the k which yields the maximum σ_B^2.

3.5.3 Segmentation of cell colonies

It soon became evident that the cell colony quantification needed to be more accurate than just the ratio of colored vs white pixels. It was desirable to find and outline the exact location and size of each colony, even if clustered together. A program was developed by Arous, D. using machine learning to characterize, extract and segment the cell colonies. The full procedure is described in "Principal component-based watershed method for image segmentation of in vitro cancer cell colonies" (submitted), which this section is based on.

Frist a principal component analysis (PCA) was done, bundling the information from the three color channels of the scan image to a single plane. Contrary to a simple grayscale image, PCA suppress various artefacts, such as contaminants or residue in the suspension medium, shadow artefacts or background noise from the scan acquisition and the cell container boundary. A k-means clustering was then performed to separate the colonies from the background. This is done by mapping the variance of the different objects in the image, as different objects have different variance. This created a binary mask with the adjacent colony components, denoted Binary Large OBjects (BLOBs).

Lastly, a topological multi-threshold watershed segmentation was applied, dividing the BLOBs into individual colonies. By multiplying the one-dimensional PCA-image with the grayscale image, a topographic surface is found. The grayscale image was first processed to remove noise. The found topographic surface reflects the colony number of each BLOB, however, noise and local irregularities in the intensity distribution might lead to over-segmentation, so an extended-minima transform was first used to morphologically find the minima regions of interest. These minima are defined by a threshold, and the choice of this threshold will vary the segmentation outcome. The optimal threshold was found using fuzzy logic to set a segmentation criterion to be maximized. An example of the outcome when applying the program on a cell flask is show in Figure 3.15.

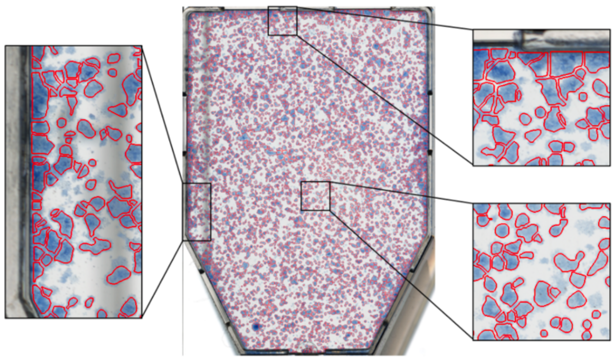


Figure 3.15: Example of segmentation on a densely populated cell flask. Segmentation done by the program is outlined in red.

3.5.4 Image registration using phase correlation

When comparing GRID irradiated films with each other, the orientation of the GRID pattern had to be similar for all films. The film receiving proton GRID irradiation were circular, which represented a problem when scanning as it was hard to accomplish a truly horizontal placement of the GRID pattern only relying on a visual verification of the orientation. Therefore, image registration using phase correlation was later applied on these films, to ensure a horizontal positioning of the GRID pattern on all films. The technique was also used on some one-dimensional curves when a match was necessary for further analyses.

With image registration, different sets of data can be transformed into one coordinate system. This is a step in the preprocessing for image analyses, using images taken at different times, viewpoints, depths, with different sensors and so on. If the images differ only in scale, rotation and/or translation, phase correlation is a possible approach for image registration. Here, the relative offset between the images is calculated using the frequency-domain representation of the data. This is normally done through Fourier transforms.

If the image f_2 is a displaced version of the image f_1 with the shift (Δx, Δy), they have the relation:

$$f_2(\mathrm{x,y}) = f_1(\mathrm{x} - \Delta\mathrm{x},\ \mathrm{y} - \Delta\mathrm{y}) \tag{3.17}$$

The Fourier transforms F_1 and F_2 of the images f_1 and f_2 respectively, is then related as:

$$F_2(\omega_1,\omega_2) = F_1(\omega_1,\omega_2)\mathrm{e}^{-2\pi j\ (\omega_1\Delta x,\ \omega_2\Delta y)} \tag{3.18}$$

The cross-power spectrum of the two Fourier transforms is defined as:

$$\frac{F_2(\omega_1,\omega_2)\ F_1^*(\omega_1,\omega_2)}{|\ F_2(\omega_1,\omega_2)\ F_1(\omega_1,\omega_2)\ |} = \mathrm{e}^{-2\pi j\ (\omega_1\Delta x,\ \omega_2\Delta y)} \tag{3.19}$$

where F_1^* is the complex conjugated of F_1. As seen from equation (3.18) and (3.19) the cross-power spectrum is equivalent to the shift of the two images. By doing an inverse Fourier transform of the cross-power spectrum, an impulse function is obtained. The peak of the impulse function – which is nearly zero everywhere else – gives the location of the displacement. This can also be used if there is a rotational or scaling discrepancy between two images. If M_1 and M_2 are the magnitudes of the Fourier transforms F_1 and F_2 respectively, and these magnitudes are converted to a polar coordinate system, we have for the rotation:

$$M_1(\rho,\ \theta) = M_2(\rho,\ \theta - \Delta\theta) \tag{3.20}$$

where $\Delta\theta$ is the difference in rotation between the two images. Similarly, if the difference is a scaling factor a, we have:

$$M_1(\rho, \theta) = M_2\left(\frac{\rho}{a}, \theta\right) \tag{3.21}$$

Combining the two, and converting into the log-polar coordinate system, we get:

$$M_1(\log\rho\,,\,\theta) = M_2(\log\rho - \log a\,,\,\theta - \Delta\theta) \tag{3.22}$$

Subsequently, by applying the phase correlation technique the scaling and rotational parameters can be determined. This is done in a manner invariant to translation due to properties of the Fourier transform. A complete procedure would therefore be to first obtain the parameters for scaling and rotation, applying them to the images and then do a second phase correlation to overcome the translational shift. The greatest strength of this method is the speed – especially for high resolution images – as a result of using fast Fourier transformations.

3.6 Statistical methods

3.6.1 Levenberg Marquardt algorithm

When selecting a model for converting netOD to dose with the gafchromic EBT-3 films, we wish to fit a variety of models to a data set. This is also true when making an LQ-model based on cell survival data from open field irradiations. In both instances, the Levenberg Marquardt algorithm is used.

A possible approach to data fitting is trough least-squares methods, where a solution is found by minimizing the sum of squared residuals (the difference between the observed and fitted value). The Levenberg-Marquardt algorithm (LM), also known as damped least-squares, has become a standard technique for solving non-linear least-squares problems. It is an iterative method, using a combination of the Gauss-Newton algorithm (GNA) and the method of gradient descent. We have an observation vector x and a model $f(\beta)$, where β is the parameter vector best satisfying the functional relation and what we wish to find. For each iteration in the algorithm, β is replaced with $\beta + \delta$, and this can be approximated with a Taylor series expansion:

$$f(\beta + \delta) \approx f(\beta) + J\delta \tag{3.23}$$

, where J is the Jacobian matrix $J = \partial f(\beta)/\partial\beta$. Denoting the sum of squares as $S(\beta)$, the minimum is achieved by requiring that its gradient equals zero. Using the approximation in equation (3.23) then yields:

$$S(\beta+\delta) \approx \left|\left| x - f(\beta) - J\delta \right|\right|^2 \tag{3.24}$$

Solving this equation, taking its derivative with respect to δ and setting the result to equal zero gives:

$$\left(J^{\mathrm{T}}J\right)\delta = J^{\mathrm{T}}[x - f(\beta)] \tag{3.25}$$

where $^{\mathrm{T}}$ is used to denote transposition. Levenberg then expanded this with a damping factor μ:

$$\left(J^{\mathrm{T}}J + \mu I\right)\delta = J^{\mathrm{T}}[x - f(\beta)] \tag{3.26}$$

where I is the identity matrix. For each iteration, the damping factor is adjusted. If μ is large, the update step δ is close to the steepest descent direction, as $J^{\mathrm{T}}J + \mu I$ is almost diagonal. If μ is small, LM is closer to GNA, and the update step is an approximation of the exact quadratic step in an ordinary least-squares case. LM accommodates its own damping – the damping is reduced unless the step fails to reduce the sum of squares, then damping is raised. This makes LM an adaptive method, capable of converging fast when close to a minimum and slow elsewhere.

LM needs an initial guess of the parameters, and in the case of several local minima, the algorithm only converges to the global minimum if the initial guess is somewhat close to a terminal solution. It can also be slow to converge, especially if the model incorporates numerous parameters. However, it is a robust method capable of handling multiple parameters and will generally find an optimal solution even when the initial guess is far off.

3.6.2 Akaike Information Criterion

Having found a selection of possible models to use for converting netOD to dose, we need a way of comparing and evaluating the goodness of these models. The Akaike information criterion (AIC) is a method for model comparison by estimating the relative quality of statistical models (both linear and non-linear) for a given data set. AIC is defined by

$$\mathrm{AIC} = 2k - 2\ln\left(\hat{L}\right) \tag{3.27}$$

where k is the number of model parameters, i.e. the number of variables in the model plus the intercept. $\hat{L}$ is the maximum value of the likelihood function for the model. For a given set of data, the preferred model will have the lowest AIC score. Thus, by equation (3.8) AIC then rewards having a high maximum log-likelihood, which is a measure of the goodness of fit of the model. Simultaneously, complicated models with increasing k are penalized and thus least favored. The AIC estimate thereby accounts for models that are prone to both under- and overfit a data set.

While AIC will yield the best model for a given set of observations, it is however incapable of establishing the overall absolute quality of this model for the data set – rather a relative assessment between the models is ascertained. The most optimal model might therefore deviate from the observations, and other evaluation methods should therefore be performed to make sure that the model actually gives a good representation of the data.

If all models in a data set assumes normally distributed errors with a constant variance, least-squares model fitting can be used. With the residual sum of squares (RSS) defined as $\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2$, y being the parameters to predict and $f(x)$ the predicted values, and the variance of the model's residual written as $\hat{\sigma}^2 = \mathrm{RSS}/n$, the maximum value of the log-likelihood function can be defined as

$$\ln\left(\hat{L}\right) = -\frac{n}{2}\ln\left(2\pi\hat{\sigma}^2\right) - \frac{\mathrm{RSS}}{2\hat{\sigma}^2} = -\frac{n}{2}\ln(2\pi/n) - \frac{n}{2}\ln(\mathrm{RSS}) - \frac{n}{2} \tag{3.28}$$

The constants $-\frac{n}{2}\ln(2\pi/n) - \frac{n}{2}$ will only increase or decrease the AIC estimate, and since it is the differences that matters with AIC, they can be ignored. AIC for least-squares model fitting can thus be written as

$$\mathrm{AIC} = 2k - n\ln(\mathrm{RSS}) \tag{3.29}$$

If the models then all have the same k, AIC reduces to a measure of RSS - which least squares model selection normally is based on.

Corrected Akaike Information Criterion ($\mathrm{AIC_C}$)

It has been shown that AIC has a tendency to choose models with too many parameters if the sample size is small compared to the number of estimated parameters. A correction for this can be done by adding a second order bias-correcting term. The corrected Akaike information criterion ($\mathrm{AIC_C}$) is given by

$$\mathrm{AIC_C} = \mathrm{AIC} + \frac{2k(k+1)}{n - k - 1} \tag{3.30}$$

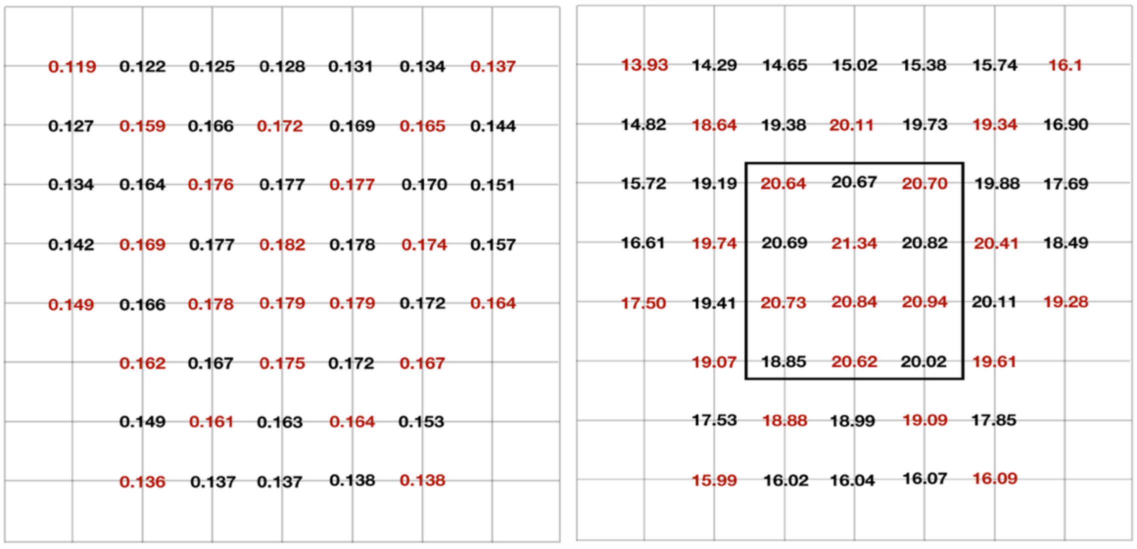
The correction term is essentially an extra penalty for the numbers of parameters in the model. For large sample sizes, the correction term converges to zero, and AIC and $\mathrm{AIC_C}$ tends to select the same model if the ratio n/k is sufficiently large. Generally, it is recommended to use $\mathrm{AIC_C}$ if $n/k < 40$. (Burnham. and Anderson.)

4 Results

4.1 Dosimetry

4.1.1 X-ray ionization chamber dosimetry

Initially, the test measurements at two random grid spots on the Perspex plate read 21.20 nC / 21.11 nC and 20.43 nC / 20.44 nC, which was deemed stable enough to continue with single measurements per spot. The current and charge detected during one minute of exposure at the Perspex plate is shown in red in Figure 4.1,where the left image shows the current, while the right image shows the charge. An interpolation was done to find the remaining points (displayed in black in the same figure). Further, a visualization of the X-ray field intensity within the irradiation chamber is shown in Figure 4.2, represented as a heatmap. Bright yellow colors correspond to higher field intensities while a low field intensity is tainted more towards darker colors. The current measured was used to determine where at the Perspex plate an area of 15x12 cm^2 received the most homogenous dose. This area as well as the determined placement of the cell flasks in the PMMA holder and calibration films are also shown in Figure 4.2. To calculate the dose rate at the chosen 15x12 cm^2 field, an average of the respective charges measured within the marked square in Figure 4.1 was employed.



0.119	0.122	0.125	0.128	0.131	0.134	0.137
0.127	0.159	0.166	0.172	0.169	0.165	0.144
0.134	0.164	0.176	0.177	0.177	0.170	0.151
0.142	0.169	0.177	0.182	0.178	0.174	0.157
0.149	0.166	0.178	0.179	0.179	0.172	0.164
	0.162	0.167	0.175	0.172	0.167	
	0.149	0.161	0.163	0.164	0.153	
	0.136	0.137	0.137	0.138	0.138	

13.93	14.29	14.65	15.02	15.38	15.74	16.1
14.82	18.64	19.38	20.11	19.73	19.34	16.90
15.72	19.19	20.64	20.67	20.70	19.88	17.69
16.61	19.74	20.69	21.34	20.82	20.41	18.49
17.50	19.41	20.73	20.84	20.94	20.11	19.28
	19.07	18.85	20.62	20.02	19.61	
	17.53	18.88	18.99	19.09	17.85	
	15.99	16.02	16.04	16.07	16.09	

Figure 4.1: Ionization chamber measurements. Values in red were measured using the chamber, values in black were found from a linear interpolation. Left image shows the current detected, and right image shows the charge measured after one minute of irradiation.

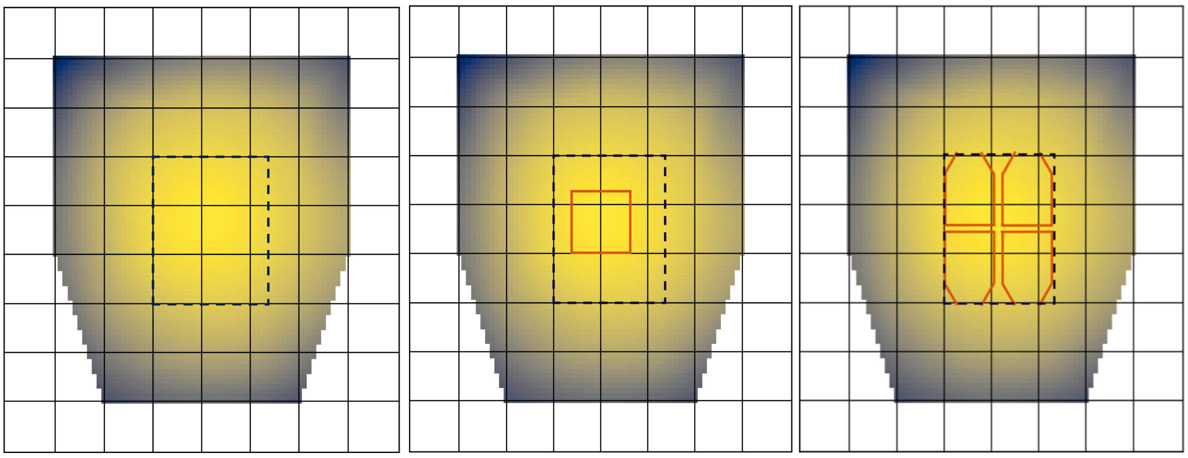


Figure 4.2: ***Left:*** *Visualization of the field homogeneity. Black dotted lines show the area chosen as the 15x12 cm^2 homogenous field.* ***Middle:*** *Placement of the Gafchromic EBT3 calibration films is shown in red.* ***Right:*** *Placement of the four T25 flasks in the PMMA holder is shown in red.*

The dose rate of the field was found using equation (3.1) with the values shown in Table 3.1. M_n is the average of the charges inside of the 15x12 cm^2 homogenous field shown inside the black square in Figure 4.1, found to be M_n = 20.653 ± 0.375 nC. The uncertainty of the dose rate was found from equation (3.3). The dose rate of the homogenous 15x12 cm^2 field is then:

$$D_{plate} = 0.500 \pm 0.010 \text{ Gy/min} \tag{4.1}$$

Additionally, a separate low dose dosimetry was done since the X-ray tube uses some seconds to reach the designated voltage. Chamber measurements were conducted with 5, 10, 15 and 20 seconds of radiation, and a linear regression was fitted to the measured points shown in Figure 4.3. The regression gives the dose as a function of time:

$$D_{plate}^{low} = (0.00865 \pm 0.00010)\, t - 0.026 \tag{4.2}$$

Where *t* is measured in seconds. Extrapolating equation (4.2) up to one minute gives a dose rate of 0.493 ± 0.006 Gy/min. This matches well with the dose rate in equation (4.1) as they are within the uncertainty of each other. Equation (4.2) is therefore used for doses up to 0.5 Gy (t < 60 sec), and equation (4.1) for doses from 0.5 Gy and above (t > 60 sec).

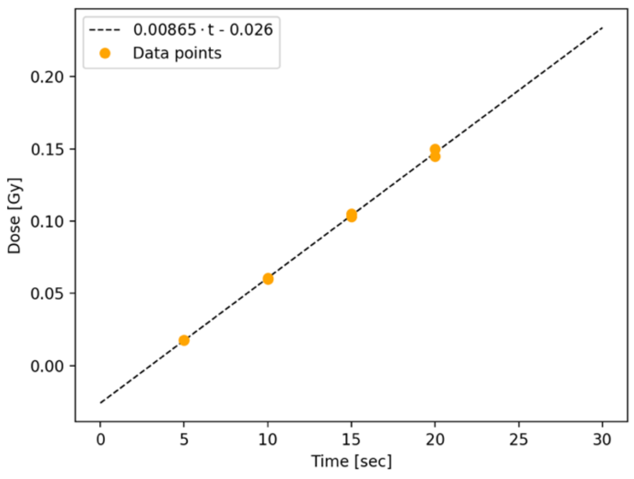


Figure 4.3: Low dose measurements on Perspex plate with a linear fit.

Chamber dosimetry was also done inside of the T25 flasks when placed in the PMMA holder. To keep track of the four positions, they were named A, B, C and D. The result of this dosimetry is shown in Table 4.1, where the dose rate was calculated from (3.1), using Table 3.1, and M_n being the average of the two chamber measurements exposed for 1 minute. The uncertainty was calculated from equation (3.3). The average dose rate over all of the four flask positions is shown in equation (4.3). As the dose rate of the different flask positions were within the uncertainties of each other, it was decided to recommend using the average dose rate regardless of which position being irradiated.

Table 4.1: Measured charges inside T25 flasks by ionization chamber and the calculated belonging dose rates.

Position	Charge 1	Charge 2	Average (M_n)	Dose rate
A	25.43 nC	25.27 nC	25.350 ± 0.080 nC	0.614 ± 0.0058 Gy/min
B	25.29 nC	25.22 nC	25.255 ± 0.035 nC	0.612 ± 0.0055 Gy/min
C	25.38 nC	25.33 nC	25.355 ± 0.025 nC	0.614 ± 0.0055 Gy/min
D	25.20 nC	25.27 nC	25.235 ± 0.035 nC	0.611 ± 0.0055 Gy/min

$$D_{flask} = 0.613 \pm 0.006 \text{ Gy/min} \tag{4.3}$$

A separate low dose dosimetry was done for the flasks as well, with the same timepoints as used on the plate and two measurements for each time point at each of the four flak positions, and the result is shown in Figure 4.4. Once again, the different positions yielded similar results, sufficient to only use the mean regardless of what flask position is irradiated. The dose rate for low doses is thus found using:

$$D_{flask}^{low} = (0.01093 \pm 0.00012)\ t - 0.035 \tag{4.4}$$

with *t* measured in seconds. Extrapolating this up to one minute gives a dose rate of 0.621 ± 0.007 Gy/min, which again falls within the uncertainty of the dose rate found in equation (4.3). Equation (4.4) is thus used for doses up to 0.5 Gy (t < 50 sec), while equation (4.3) is used for doses from 0.5 Gy and above (t > 50 sec).

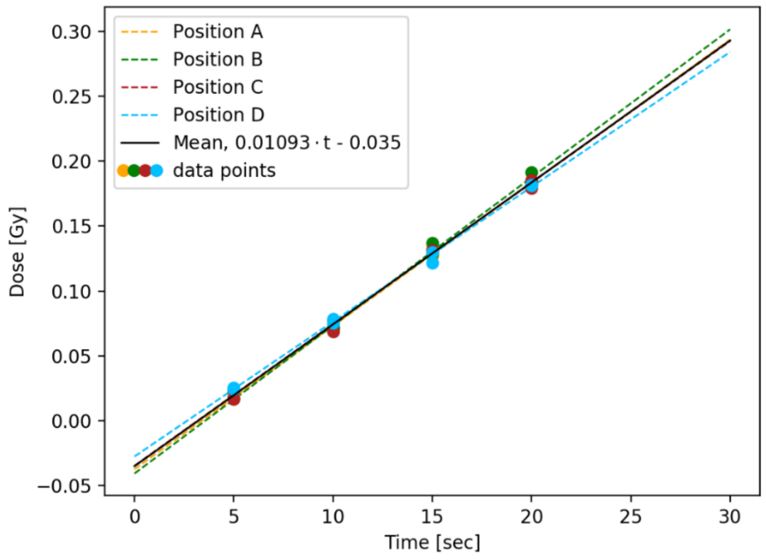


Figure 4.4: Low dose measurement in T25 flasks with linear fits.

4.1.2 Calibration of Gafchromic EBT-3 films

Two separate experiments to calibrate the films were done, and they will be referred to as experiment 1 and experiment 2. The average of the unprocessed pixel values from the green color channel from a centered ROI of 4x4 mm^2 in all calibration films are shown in Figure 4.5. It was chosen to do one common calibration using the calibration films from both experiments.

Using a ROI of 4x4 mm^2, the netOD was extracted for all four color channels, seen in Figure 4.6. A verification of the choice of ROI size was later done, using a ROI of 4x4 mm^2 and 3x3 mm^2 placed at the center of the films. The relative error in the doses produced by the two different ROI sizes was estimated to be less than 0.01% on average, where both film calibrations maintained the same ranking order of the candidate film models, but with an improved goodness of fit estimates when using a ROI size of 4x4 mm^2. Because of this, a 4x4 mm ROI was chosen for the calibration. Moreover, as some of the calibration films had a few scratches (by which could invoke signal artefacts), selecting the larger ROI size would possibly average out these inaccuracies and give trustworthy results.

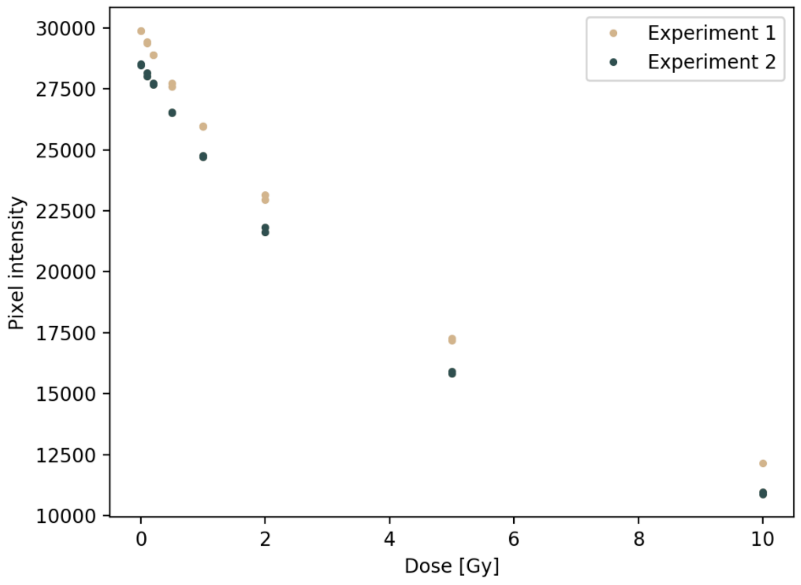


Figure 4.5: Pixel intensity vs delivered dose to calibration films extracted from the green color channel, including the unirradiated control films.

Before testing the different models established in section 3.1.1, it was necessary to establish the parameter *n* in model 1 and model 4 (shown in Table 4.2). These models were tested with *n* ranging from 0 to 5 with steps of 0.25. The corresponding R^2_{Adj} was found for all steps, and the *n* giving the highest R^2_{Adj} was extracted for each channel, shown in Table 4.3. By attaining the value for *n* in which optimized the R^2_{Adj} for the respective channels, it was decided to keep this parameter fixed throughout the regression analysis and subsequent model selection. Four dissimilar models, shown in Table 4.2, served as candidates for film calibration and were submitted for goodness-of-fit. evaluation, where R^2_{Adj}, the parameters p-values (both relevant solely for the linear models), AIC_C and RSS benchmarks are tabulated in Table 4.4. The best fit to the models as well as their residuals and uncertainties are shown in Appendix A.

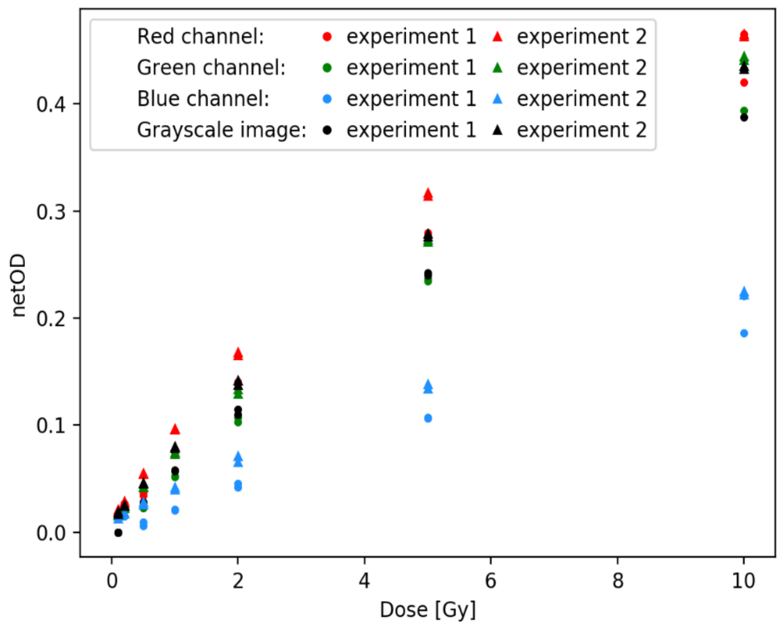


Figure 4.6: Measured netOD for all color channels of all calibration films.

Table 4.2: Models tested.

Model 1*	$a \cdot \text{netOD} + b \cdot \text{netOD}^n$
Model 2	$a + \dfrac{b}{\text{netOD - } c}$
Model 3	$a \cdot \text{netOD} \cdot \text{e}^{\,b \cdot \text{netOD}}$
Model 4*	$a \cdot \text{netOD}^n$

*n for used for the different color channels are shown in Table 4.3.

Table 4.3: Best n found for each color channel, with the belonging r^2 value when using this n.

	Model 1		**Model 4**	
Channel	***n***	$\mathbf{R^2_{Adj}}$	***n***	$\mathbf{R^2_{Adj}}$
Red	2.00	0.9896	1.50	0.9887
Green	1.50	0.9903	1.25	0.9903
Blue	0.75	0.9767	1.25	0.9751
Grey	1.50	0.9897	1.25	0.9985

Table 4.4: Comparison of models.

Model	**Channel**	**AIC$_C$**	**RSS**	**R^2_{Adj}**	**p-values**
Model 1	Red	38.00	3.58	0.9888	*a:* 6.4e^{-5} *b:* 2.8e^{-8}
	Green	35.97	3.08	0.9896	*a:* 3.9e^{-4} *b:* 9.4e^{-8}
	Blue	61.67	7.94	0.9737	*a:* 8.7e^{-7} *b:* 1.2e^{-2}
	Gray	37.76	3.28	0.9889	*a:* 8.8e^{-2} *b:* 4.6e^{-6}
Model 2	Red	43.26	3.65	-	
	Green	40.59	3.32	-	
	Blue	66.59	8.40	-	
	Gray	42.69	3.58	-	
Model 3	Red	39.01	3.43	-	
	Green	37.64	3.27	-	
	Blue	64.16	8.43	-	
	Gray	39.38	3.48	-	
Model 4	Red	38.56	3.67	0.9883	*a:* 1.4e^{-27}
	Green	34.03	3.12	0.9900	*a:* 8.6e^{-28}
	Blue	60.15	7.96	0.9742	*a:* 1.7e^{-22}
	Gray	38.04	3.60	0.9880	*a:* 7.2e^{-27}

In view of the R^2_{Adj} and AIC$_C$ scores, the model with the highest R^2_{Adj}, lowest AIC$_C$ estimate and also the lowest p-value, was provided by model 4 when adopting the green color channel. An evaluation of the residuals and uncertainty of the different models were also done – the latter is particularly important when employing the model on different films later as we want the conversion to be as accurate as possible. Thus, the conversion from netOD into dose is governed by the following model, adopting the green color channel:

$$D(\text{netOD}) = a \cdot \text{netOD}^{n} = 28.358 \cdot \text{netOD}^{1.25} \tag{4.5}$$

a (= 28.358) here have a standard deviation of 0.424 and the p-value shows high statistical significance (Table 1.4; $p \leq 0.05$). A plot of this model together with the netOD points used for calibration is shown in Figure 4.10. Figure 4.9 shows the belonging residual plot with trend curve, whereas Figure 4.8 shows the associated uncertainty with dose originating from the fit itself, the experimental uncertainty and the total propagated uncertainty as they are defined in

Appendix A. Since the total uncertainty is only valid for the discrete netOD estimates used in the calibration, a spline interpolation was done in order to make it a smooth function of netOD. Sampling the interpolation with 10000 points, a look-up table was made where the uncertainty for doses between 0.1 and 10 Gy could be found. The interpolation curve is shown in Figure 4.7.

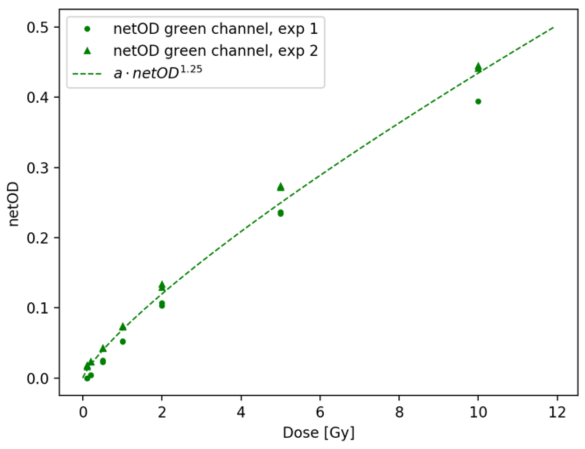


Figure 4.10: Best fit to calibration films.

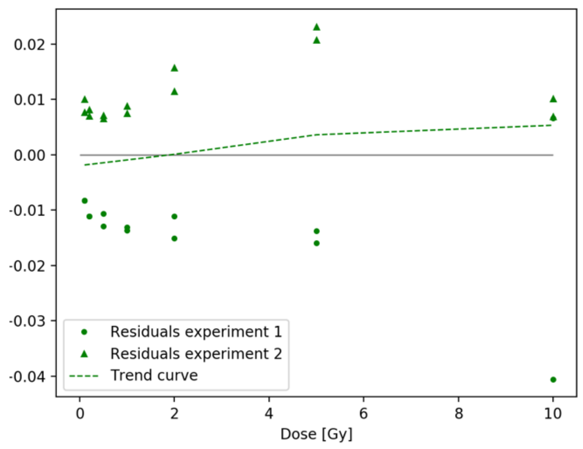


Figure 4.9: Residual plot and loess trend curve for the best fit.

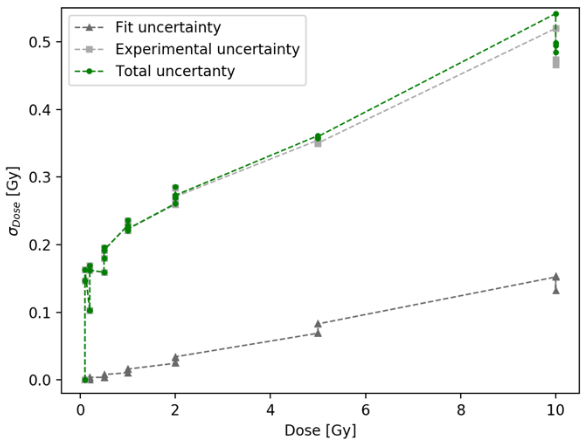


Figure 4.8: Uncertainty plots of the best fit.

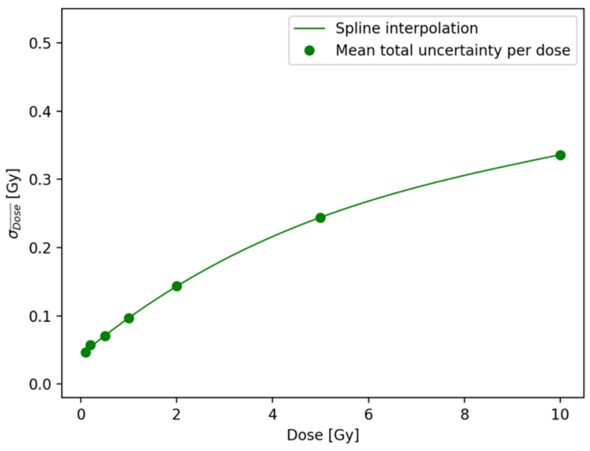


Figure 4.7: Average uncertainty with spline interpolation.

4.1.3 X-ray GRID films

Having established a conversion from netOD to dose, it was possible to map the dose within the T25 flaks when irradiated with GRID. Two independent experiments were conducted for film irradiation with GRID, where both of them were conducted in conjunction with the calibration experiments. Thus, experiment 1 now refer to the GRID irradiations done simultaneously as calibration experiment 1, and similar for experiment 2.

An example of an unprocessed irradiated film is displayed to the left in Figure 4.11, while the right shows the same film converted into dose using the model established from the calibration. As the unprocessed film shows, the left edge of the films has some artefacts inherited from the cell flask edges due to the scanning procedure with the flatbed scanner. The films also have various degrees of damage at the other edges originating from cutting the films to their shape, and there was a piece of tape on one end during radiation to enable insertion and extraction of the films into and out of the flask. This is cropped from all films to avoid it influencing the results.

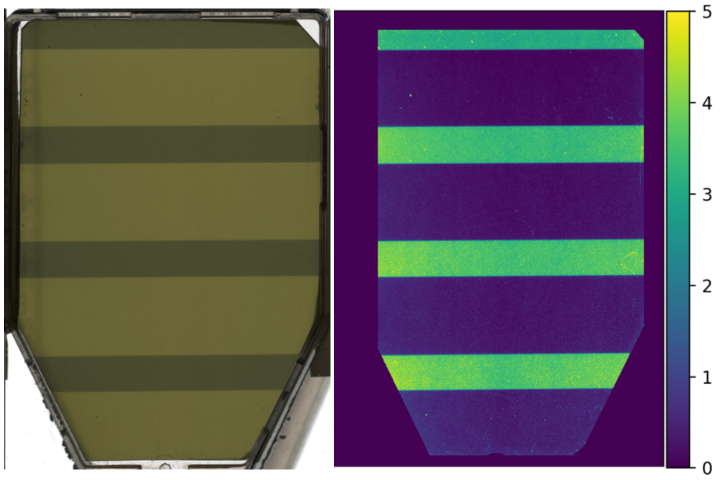


Figure 4.11 : Example of dose deposition on EBT3 films when irradiating with GRID. ***Left:*** *Unprocessed film inside of T25 flask.* ***Right:*** *Edges cropped away and film converted to dose using equation (4.5) . Bar shows Gy.*

The average unprocessed pixel intensity drawn across each film, eight from each experiment, is shown in Figure 4.12, all profiles are always adjusted for the shape of the films. An important observation to be made is that the pixel intensities of the GRID films from experiment 1 are all lower than experiment 2, whereas the opposite intensity order is given from the film calibration experiments. One of the profiles in Figure 4.12 clearly diverge from the rest and was regarded as an outlier and omitted from further analysis.

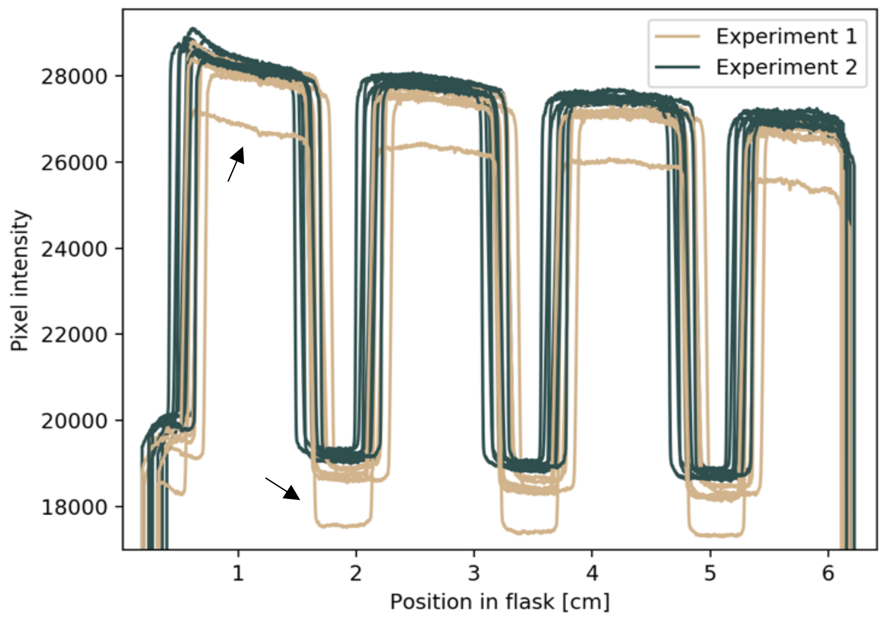


Figure 4.12: Average pixel intensity found with the green color channel per pixel row for all X-ray GRID films. Arrows point towards the one outlier profile which was excluded from further analyses.

In addition to irradiating films with GRID, each experiment also included two films exposed to an open field (i.e. in T25 flasks in the PMMA holder) with the same noiminal dose (5 Gy) as the films irradiated with GRID. This was performed in order to verify the attributes of the peak dose across the profile with respect to an open-field irradiation. Dose profiles from all of the GRID films as well as dose profiles from these open field irradiations are shown in Figure 4.13. There is a distinct difference in dose from the two open-field irradiations, and also a difference between the peak dose and open-field irradiation for the two experiments. Looking at the GRID profiles, there are clearly irregularities both in positioning and dose response, however, the discrepancies in position are due to uncertainties in the film set-up and not the dosimetry itself. For instance, the films can move slightly inside of the flask both when irradiating and scanning, and there is also some space between the cell flasks and the PMMA cell flask holder. The shifts in position are therefore not representative of the dose delivery to the cell flasks, and a translational match is done, shifting all films to match the median film profile.

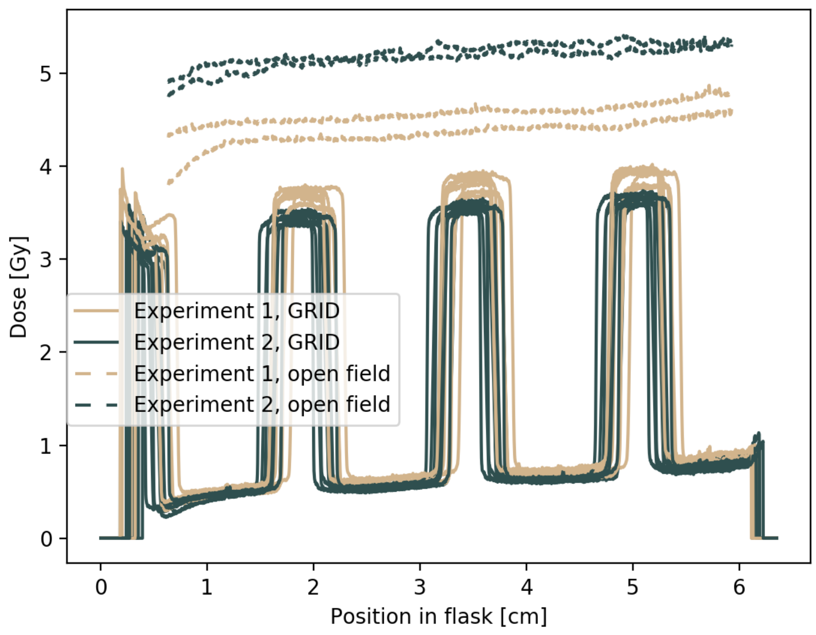


Figure 4.13: Dose profiles from all X-ray GRID films, as well as dose profiles from films irradiated with the same set-up in an open field. All films were given 5 Gy.

Figure 4.13 also shows slight increase in dose as we move through the flask. This is not a dosimetric property as the increase in apparent dose is toward the more distal ends of the field. It turns out that when scanning the films, the scanner lid could not be closed properly as the films were scanned inside of the cell flaks, leaving the scanner lid in a tilted position. This projected a gradient across the film and therefore the pixel values, making them higher where the lid is closest and lower as the distance to the lid increases. This phenomenon can also be observed when extracting profiles from the calibration films, which are certain to have received a homogenous dose (see Figure 4.2). The dose profiles from calibration experiment 2 can be seen in Figure 4.14.

To correct for this gradient, trend curves were established by doing linear regression on the dose profiles of all calibration films given 5 Gy. These four films, two and two derived from each calibration experiment were chosen because they were given the same nominal dose as all films irradiated with the GRID set-up. The calibration films only cover around half of the cell flask, but we assumed the gradient to be linear, and the trend curve is thus valid when extrapolated to cover the entire flask. The difference from each of these trend curves to a baseline of 5 Gy was found for each pixel row, and the mean of the four difference vectors was used as a correction vector. The correction was then applied to all films irradiated with the GRID set-up, and their profiles after being corrected are shown in Figure 4.15.

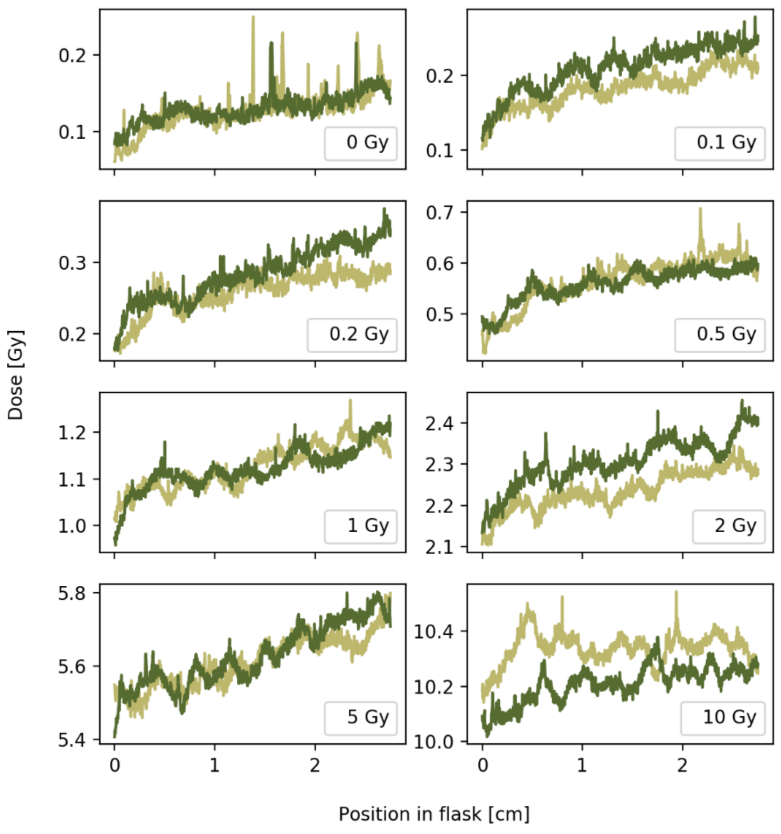


Figure 4.14: Dose profiles from all calibration films from experiment 2, two films per dose. A clear increase in dose can be seen in all films, indicating this gradient arose in the scan.

Using the corrected version of the GRID profiles, the mean dose profile for both the GRID-films and the open-field films are found. A 95% confidence interval (CI) was also found for both, taking both the uncertainty of the netOD to dose conversion, $\sigma_{D_{fit}}$ found from the spline interpolation, as well as the standard deviation of the films, σ_{GRID}, into account. This is done using the equation:

$$CI_{95\%} = 1.96\sqrt{\sigma^2_{D_{\text{fit}}} + \frac{\sigma^2_{\text{GRID}}}{\sqrt{N}}} \tag{4.6}$$

where N is the number of GRID films. The result can be seen in Figure 4.16. The irregularity at the edges are solely from cropping the films and are thus not representative of the dose at these positions and was therefore not evaluated. Recalling that all films were originally given a nominal dose of 5 Gy and thereafter appropriately scaled up/down in dose, the average dose received in the peaks and valleys of the GRID as well as the average dose received with the open field is shown in

Table 4.5. The high uncertainty seen in the open field profile mainly stems from the high standard deviation in the films between the two experiments.

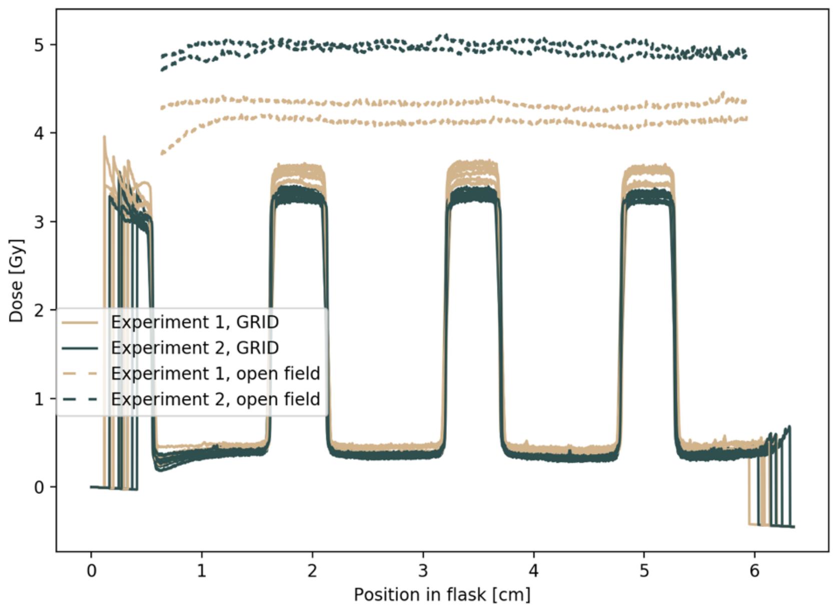


Figure 4.15: Dose profiles from X-ray GRID films and films irradiated with the same set-up in an open field. All films were given a nominal dose of 5 Gy. The profiles have been corrected for the scanning gradient and shifted horizontally to match the median GRID film.

Table 4.5: Average dose to film X-ray

Open field	Peak dose GRID	Valley dose GRID
4.59 ± 0.64 Gy	3.41 ± 0.37 Gy	0.38 ± 0.17 Gy

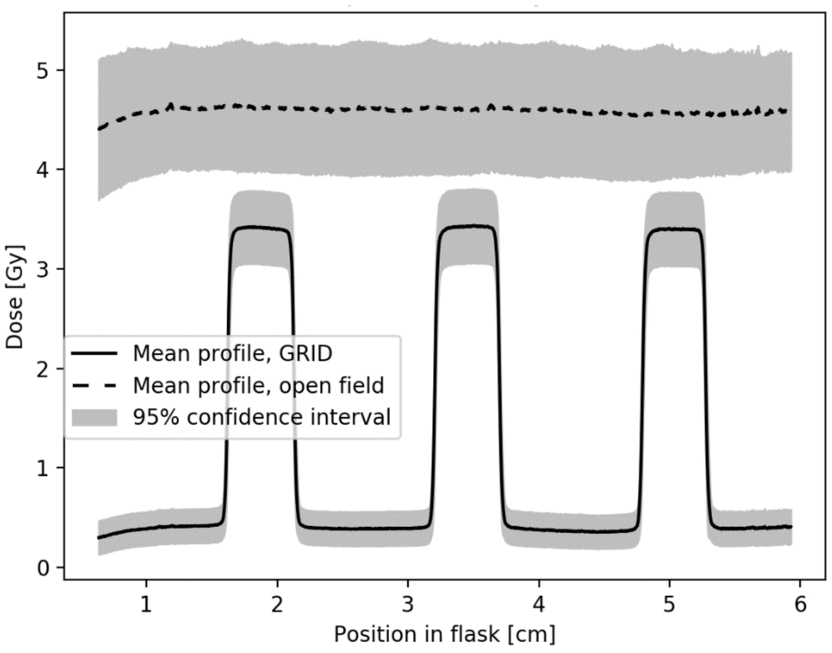


Figure 4.16: Mean dose profile from X-ray GRID films and mean profile of the films irradiated with the same set-up in an open field. All films were given a nominal dose of 5 Gy. The mean is found using the corrected and horizontally matched profiles and the confidence intervals from equation (4.6).

4.1.4 Proton film dosimetry

Two experiments were done irradiating EBT3 films with protons and will be referred to as experiment 1 and experiment 2, not to be confused with the X-ray experiments. The distances and factors used to calculate the monitor units (MU) required to give a certain dose are shown in Table 4.6. As it was difficult to time the stopping of the beam precisely, monitor units actually delivered was noted for each film. The true delivered doses are shown in Table 4.7, with two films for each dose. For the GRID films, this is the dose they were irradiated with, and not necessarily what they received under the GRID.

Table 4.6: Distance between EBT3 film and beam exit window as well as the factor used to calculate MU's (equation (3.9)) for the two experiments.

	Experiment 1	Experiment 2
Distance to beam exit window	107.5 cm	108.4 cm
Factor to calculate MU	0.894	0.902

Table 4.7: Nominal doses vs the doses actually delivered due to difficulties of timing the stopping of the beam precisely.

	Experiment 1	Experiment 2
Nominal dose [Gy]	Delivered dose [Gy]	Delivered dose [Gy]
2	2.01	2.04
	2.09	2.05
5	4.99	5.02
	5.12	5.04
10	10.04	9.93
	10.06	9.97
5 (GRID films)	5.01	5.04
	5.05	5.01

We wished to compare the films irradiated with protons to the films irradiated with X-rays, but since the scanning procedure was different for the two this could not be done directly. However, the X-ray films were originally scanned with the same protocol as the proton films before a protocol enabling a one-to-one comparison with the cell experiment was made. Consequently, the X-ray films were re-scanned with this new protocol. Looking at the measured netOD of the same X-ray films using the two different scan protocols shown in Figure 4.18, there is a distinct difference, and using the calibration found in equation (4.5) on the proton films will thus give erroneous results. However, having scans of the X-ray films with both protocols made it possible to do calibration of the old scans to match the new, and the calibration done on the old X-ray scan could consequently be applied to the proton films. The calibration was done completely equivalent to the original calibration: Using a ROI of 4x4 mm^2 and a weighted mean of both the unirradiated and background films from each experiment which then were calibrated to model 4 ($a\ netOD^{1.25}$) adopting the green color channel. To verify the reliability of this new calibration, the old X-ray scans using this calibration and the true X-ray scans using the original calibration (equation (4.5)) was plotted together, shown in Figure 4.17.

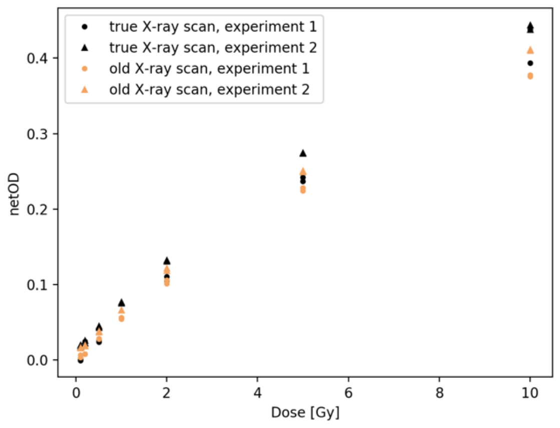


Figure 4.18: netOD from green channel for the same X-ray calibration series scanned with two different protocols.

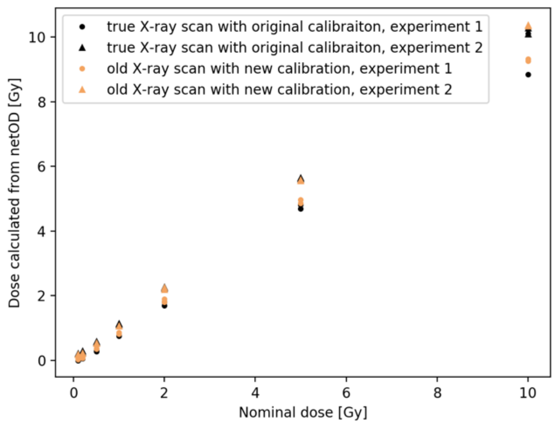


Figure 4.17: Film scanned with the correct (true) scan protocol, converted to dose using equation (4.5), and the same films scanned with the old protocol converted to dose using equation (4.7).

The relative difference of the dose calculated with the two different calibrations lies between 0.3% and 5.8%, and as the new calibration gives doses lying in between the doses found with the original calibration it is decided to proceed with this new calibration on the proton films. The model is shown in equation (4.7). A plot of both the experimental, fit and total uncertainty is shown in Figure 4.20 while Figure 4.21 is the total uncertainty of the average netOD per dose found equivalently as for the original calibration. This average uncertainty is interpolated with a spline, sampled with 10000 points in order to make a smooth look-up table of the uncertainty when converting netOD to dose. This model was then employed on all proton films, and Figure 4.19 shows the open field proton films together with the calibration X-ray films given the same doses.

$$D(\text{netOD})_{\text{proton}} = 31.477 \cdot \text{netOD}^{1.25} \tag{4.7}$$

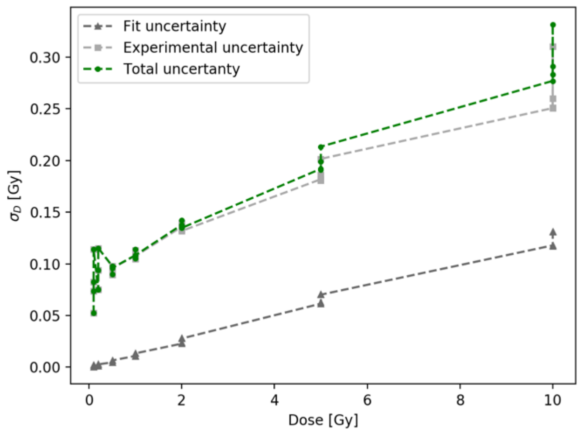


Figure 4.20: Total uncertainty, experimental uncertainty and fit uncertainty, all per dose given.

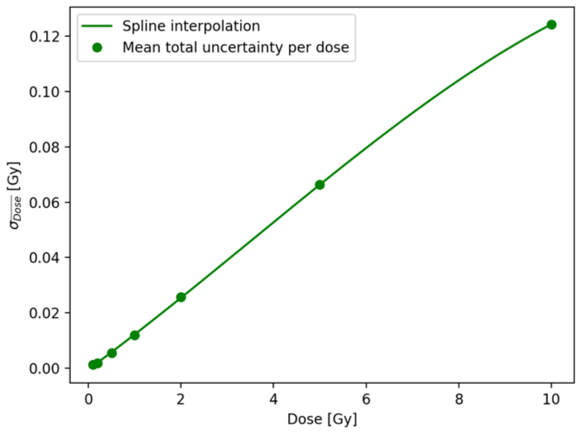


Figure 4.21: Spline interpolation to fit the total uncertainty of the average netOD given to the films.

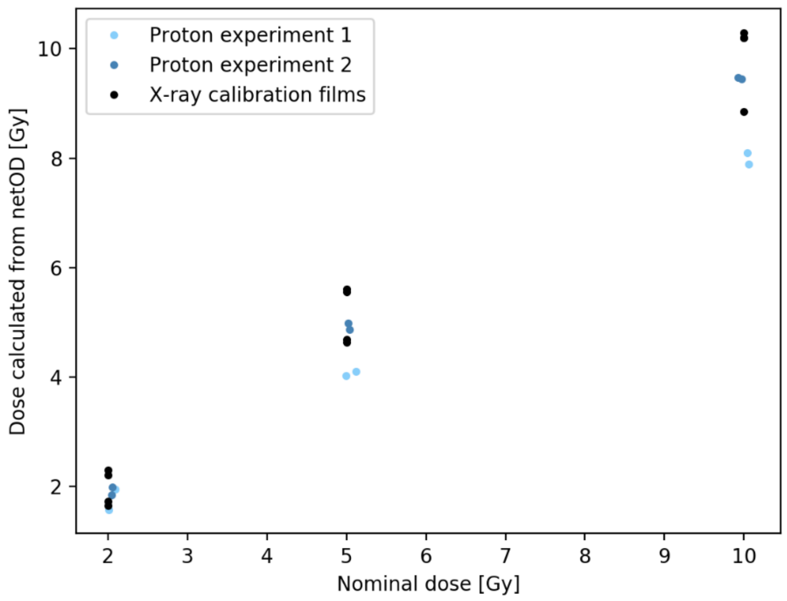


Figure 4.19: Open field irradiation with protons compared to X-rays. Protons films are plotted against the actual given doses from Table 4.7.

Looking at the unprocessed proton GRID films, shown in Figure 4.22, the stripes of the GRID are not completely horizontal for all films. This is because the circular films are unintentionally rotated during the scanning. When extracting a vertical profile through these films, the gradient of the transition from shielded to irradiated parts of the GRID will then be lower due to this rotation. To avoid this and get "true" gradients, a phase correlation was done using film B (Figure 4.22) as a template as this was considered truly horizontal. The phase correlation also translationally aligned the films in terms of the GRID pattern.

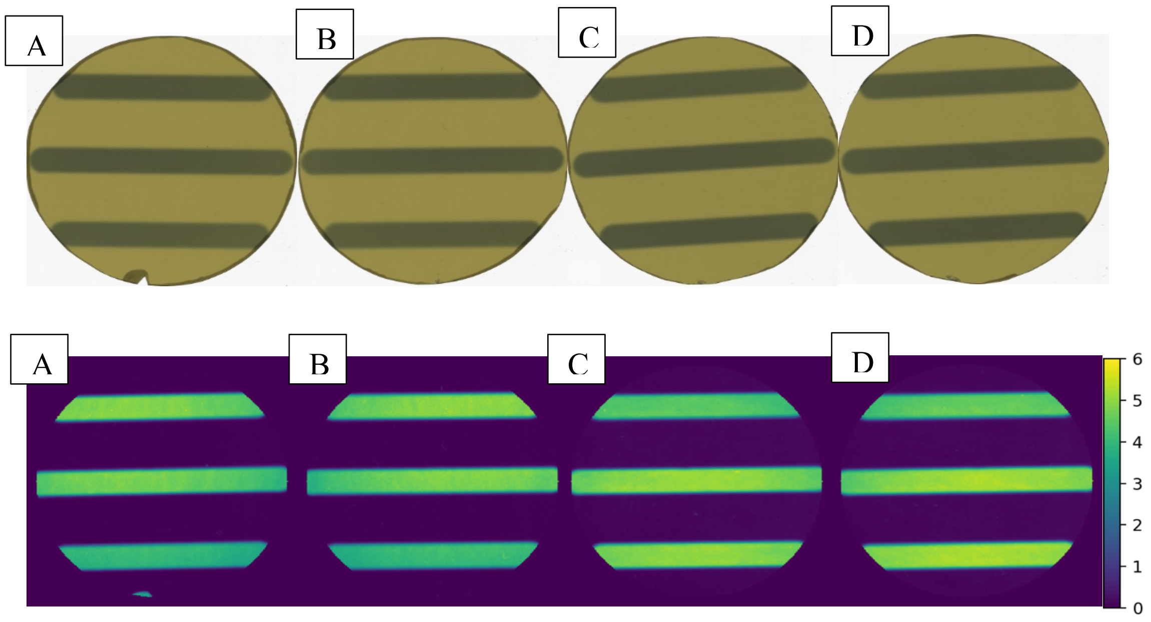


*Figure 4.22: **Top**: The four unprocessed proton GRID films. Film B was used as a template to match the other films to, both for rotation and translation. **Bottom:** Same films as above after phase correlation, cropping of the edges and also converted into dose using equation (4.7). Bar shows Gy.*

The films were both rotated and translated to match this film. Similar to the X-ray films, the edges were cropped off to avoid them influencing the signal. The films after being corrected, cropped and converted into dose are also shown in Figure 4.22. The dose profiles were subsequently obtained from the GRID films together with profiles from the open field films receiving a nominal 5 Gy dose, which is shown in Figure 4.24. The mean dose profile was found for both the GRID and open field films, together with a 95% CI taking both the uncertainty in the dose conversion as well as the standard deviations of the films into account, as shown in equation (4.6). The mean profiles and confidence interval are show in Figure 4.23, while Table 4.8 shows the average dose received at the peaks and valleys compared to the open field films.

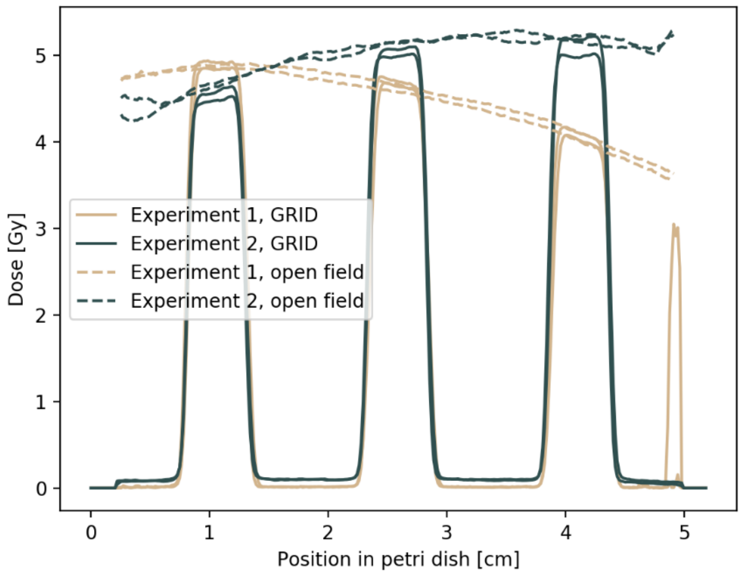


Figure 4.24: Dose profiles from all proton GRID films, as well as dose profiles from films irradiated with the same set-up in an open field. All films were given 5 Gy.

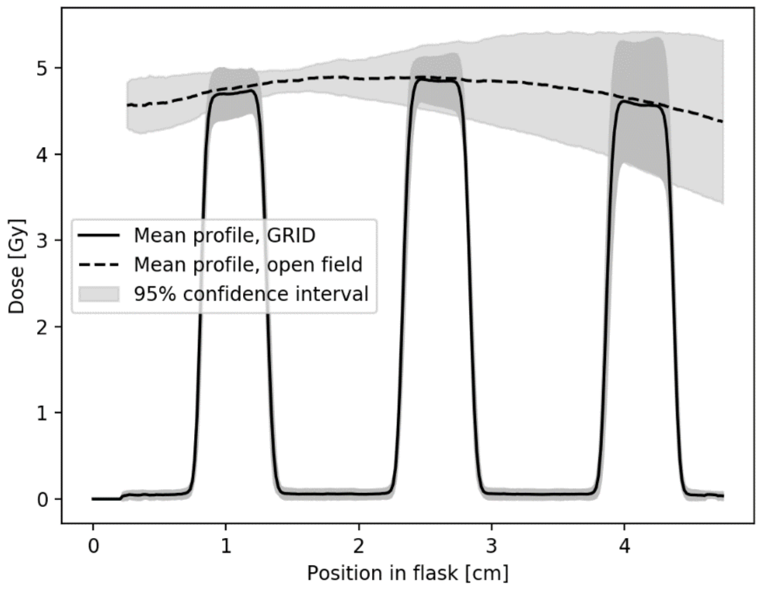


Figure 4.23: Mean dose profile from proton GRID films and mean profile of the films irradiated with the same set-up in an open field. All films were given 5 Gy.

Table 4.8: Average dose to film proton

Open field	Peak dose GRID	Valley dose GRID
4.66 ± 1.04 Gy	4.71 ± 0.43 Gy	0.06 ± 0.06 Gy

The GRID dose profiles for both X-rays and protons are shown in Figure 4.25. The X-ray profile was interpolated down to match the proton profile, as the resolution was 1200 dpi for the X-ray scan but only 150 dpi for the proton. A translational shift was also carried out to match the profiles. The proton profile has a less steep gradient between the peak and valley doses, as well as both higher peak doses and lower valley doses.

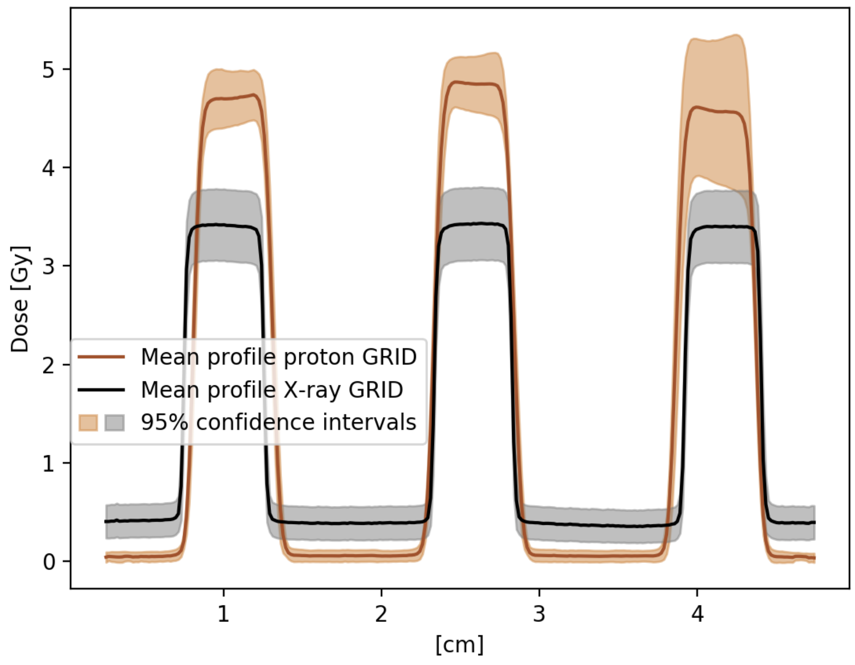


Figure 4.25: Mean dose profile from both the X-ray GRID films and proton GRID films. X-ray profile was interpolated down to match the resolution of the proton profile, and the profiles match using phase correlation.

4.2 FLUKA Monte Carlo simulations (X-rays)

A FLUKA Monte Carlo (MC) simulation imitating the GRID irradiation experiments with X-rays was done to verify the Gafchromic GRID film dosimetry. The MC simulation provides an energy deposition expressed in GeV per cm^3 per primary mass. This was obtained for each of the four flask positions is the PMMA holder, but as the simulated field, and thus the underlying dose pattern, was symmetrical over the T25 flasks, they were flipped with respect to each other and aggregated together. This was performed before extracting the average dose profile of the MC simulated dose maps. In order to get a meaningful comparison between the experimental dose maps from the film dosimetry and the MC dose maps, both profiles were normalized with respect to the median peak dose. Furthermore, spatial resolution between the two differed, as the films were scanned with a resolution of 1200 dpi whereas the MC maps were scored with 400 x 400 rectangular bins. Hence, the dose profile from the film dosimetry was interpolated down to match the MC data. This was done get a point-by-point comparison, and to be able to extract the relative differences between the profiles. The result is shown in Figure 4.26.

Peak to valley ratios are shown in Table 4.9 and was found using the median value of all peaks and all valleys, as the MC data are quite noisy. As Figure 4.26 shows there is no significant difference between the two, although here is a tendency of the EBT3 films to give higher valley doses, on average around 20% higher than from the MC simulation.

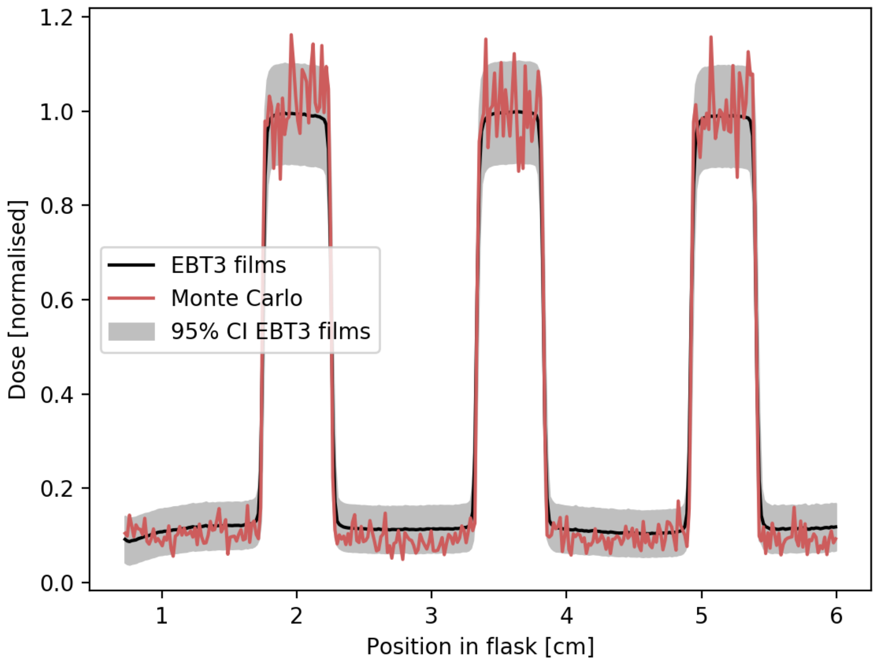


Figure 4.26: Normalized dose profile from Monte Carlo simulation of GRID irradiation together with normalized dose profiles found from EBT3 film dosimetry.

Table 4.9: Peak/valley ratios of the dose deposition from MC and EBT3 dosimetry.

	Monte Carlo	EBT3 films
Peak/valley ratio	10.4	8.1

4.3 Quantification of cell survival

The examples in this section are from X-ray cell experiments, however, the same methods were applied to the proton cell experiments. Examples of cell flaks irradiated with 5 Gy in an open field and with the GRID is shown in Figure 4.28 as well as one of the control flasks from the same experiment. Figure 4.27 shows flasks irradiated with nominal doses of 2 Gy, 5 Gy and 10 Gy, all with GRID.

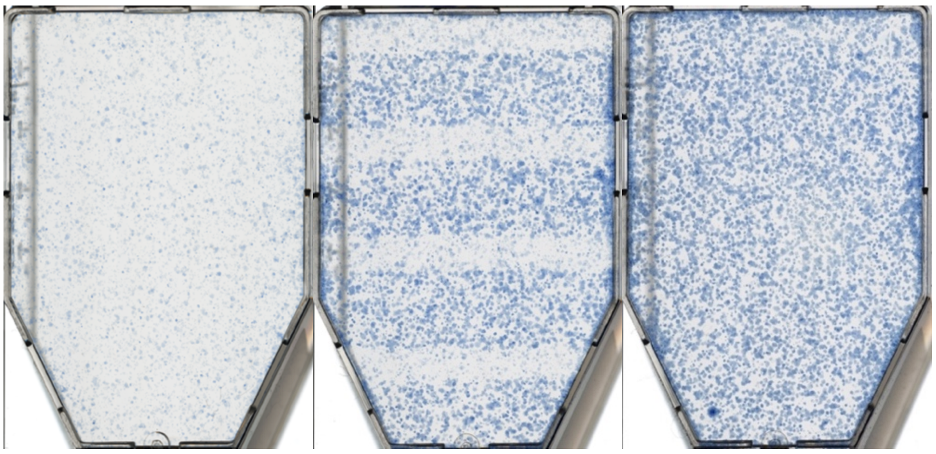


Figure 4.28: Cell assays. Left flask was irradiated with 5 Gy in an open field, middle flask was irradiated with 5 Gy using GRID in the same position as the left flask, right flask is an unirradiated control. All are from the same experiment.

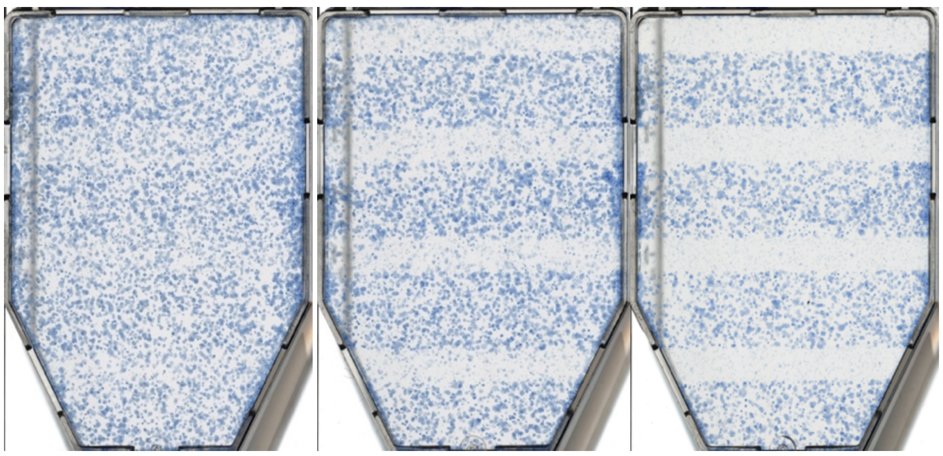


Figure 4.27: Examples of flasks irradiated with GRID. Left flask was given 2 Gy, middle flask was given 5 Gy, right flask was given 10 Gy. All are from the same experiment.

As we had dose profiles from the GRID film dosimetry, a comparison to the cell survival profiles with GRID was warranted. The first strategy was to simply use the relation between colored pixels and background, by first filtering and thresholding the images of the cell flasks. An example of this procedure is shown in Figure 4.30. Having a binary image where cells was separated from the background, a profile was drawn though the flask, and the percentage of the flask being covered by cells for each pixel row was found, shown in Figure 4.29. However, this turned out to be an inadequate representation of the cell survival, as it still included dead cells and not gave any information on the number of viable colonies, the latter being a disadvantage as both plating efficiency and surviving fraction is defined based on the number of colonies in a cell flask. This quantification method was therefore not used for any further analyses.

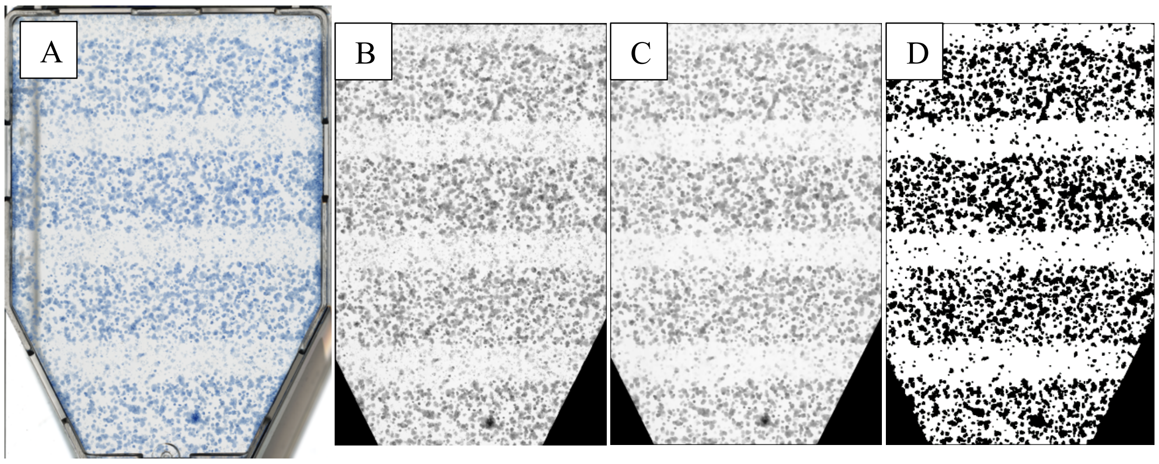


*Figure 4.30: **A:** original scan of cells irradiated with 5 Gy GRID. **B:** Edges cropped away and image converted to grayscale. **C:** Morphological opening using a circular structure element with radius of 10 pixels. **D:** Optimal thresholding using Otsu's method, giving a binary image.*

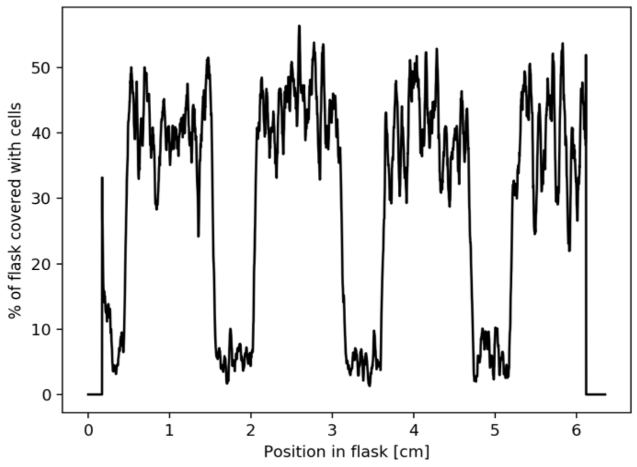


Figure 4.29: Profile of the percentage of flask covered with cells.

The method used for cell quantification is the colony counter program developed by Arous, D. An example of the flasks run though the program is shown in Figure 4.31, the cell flasks used here are the same as in Figure 4.28. The colonies detected are outlined in red, and the program provides the centroid coordinates of all colonies, among others. However, just drawing the profile through the entire cell flask by counting all colonies centered at each pixel row would give extremely noisy results. To overcome this issue, the flasks were divided into horizontal bands of thickness Δx. If the center of a colony was placed within this band, it was counted. The colony size was thus ignored, and every colony was only counted once. The counting was adjusted for the number of plated cells and for the shape of the flask, as the area becomes narrower towards the tip. Different Δx's was tested to find an optimal band width, illustrated in Figure 4.33. It was decided to use both a Δx of 0.5 mm and 1.5 mm to see whether the choice of band width impacted further results.

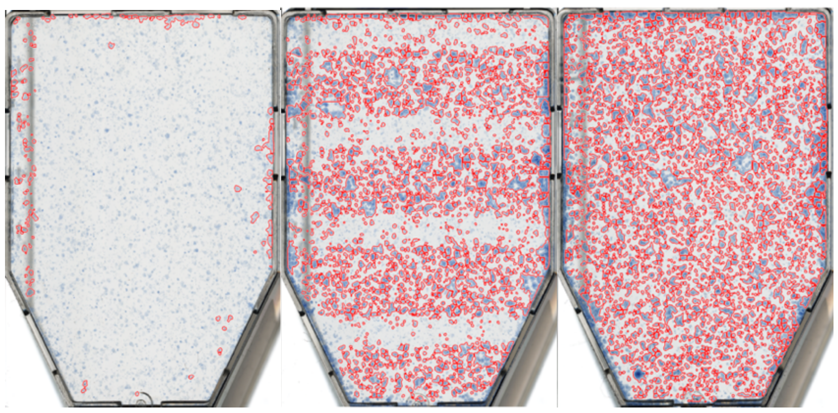


Figure 4.31 The same cell flasks as in Figure 4.28, run through the colony counter algorithm. Detected cell colonies are outlined in red.

Having established a way to acquire cell count data from the cell experiments, Figure 4.32 shows an example of cells irradiated with GRID together with cells irradiated in an open field exposed to the same nominal dose at the same flask position from the same experiment, along with the averaged pool count from the control experiments. It indicates that collimated parts of the GRID results in similar survival as the controls, while irradiated parts of the GRID give a survival similar to the open field irradiations.

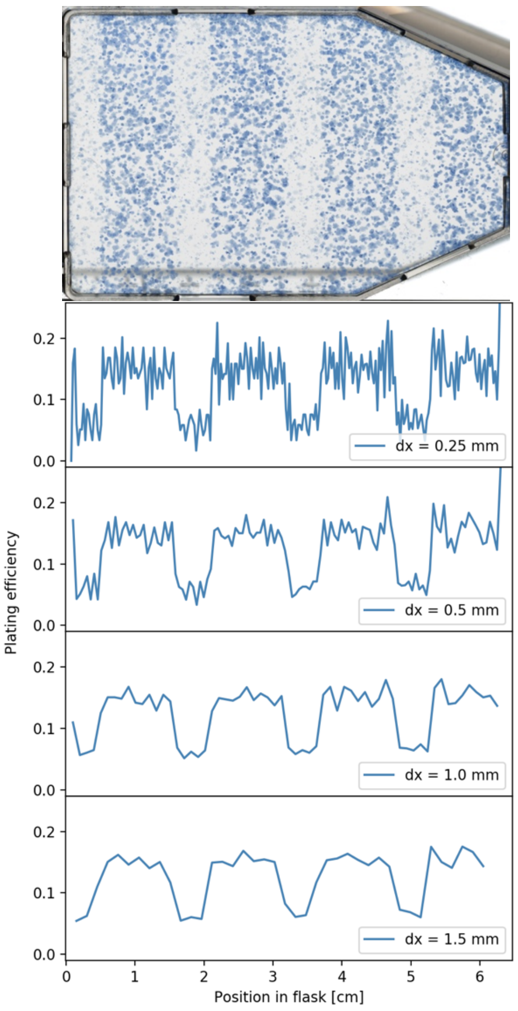


Figure 4.33: Counting colonies within a band, testing with different band widths, Δx.

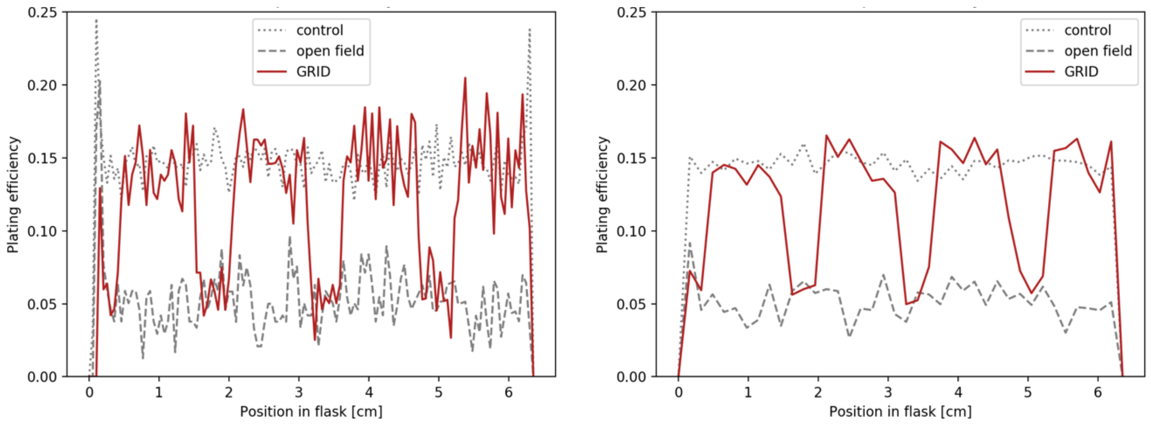


Figure 4.32: Example the cell survival after GRID irradiation, open field irradiation and a control flask. Both GRID and open field were irradiated at the same position and given 5 Gy. All from the same experiment. ***Left:*** *Counted using a band width of Δx = 0.5 mm.* ***Right:*** *Counted using a band width of Δx = 1.5 mm.*

A comparison of the response from the two different GRID cell experiments is shown in Figure 4.34, where two cell flasks irradiated in the same position with the same dose but from the two different cell experiments are used. This was done to see the potential difference in cell response from the different experiments, and also if that discrepancy disappeared after normalization. As the figure suggests, the two experiments yielded similar results even before normalizing, indicating good reproducibility of the cell experiments.

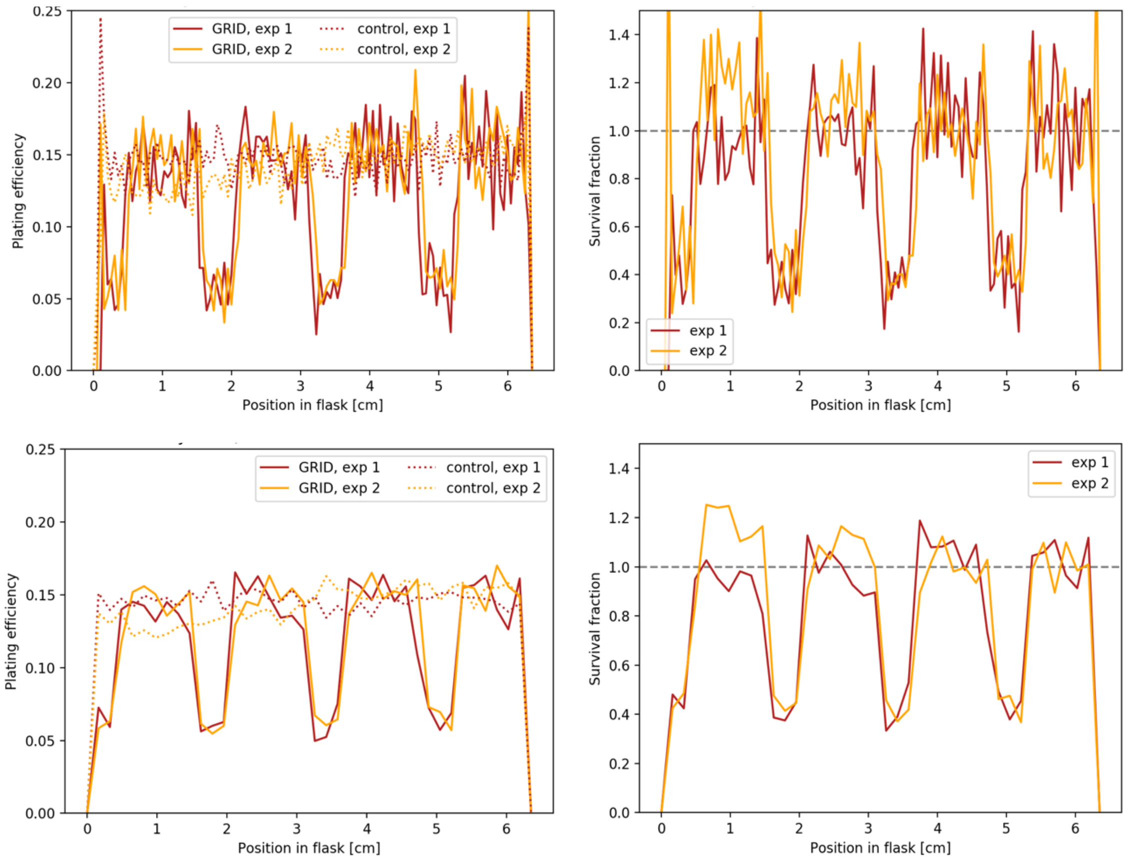


Figure 4.34: Two flasks from two different experiments, both irradiated with nominal 5 Gy GRID at the same position. ***Left:*** *GRID flasks vs the average of the controls from the same experiment.* ***Right:*** *GRID flasks are normalized to the average control, shows the survival fraction for the two GRID flasks.* ***Top:*** *Counted using a band width of 0.5 mm.* ***Bottom:*** *Counted using a band width of 1.5 mm.*

4.4 Assessment of dose distribution versus cell survival with GRID

4.4.1 LQ modelling

The dose distribution in the cell flasks when irradiated with GRID is known from the film dosimetry. With an LQ-model for cell survival as a function of dose, this dose distribution can be used to make a prediction of the cell survival undergoing GRID irradiation, which then can be compared to the actual survival observed from the cell experiments. As X-ray cell experiment were done with GRID and open fields, the survival observed at the open field irradiations can be used to make an LQ curve predicting the survival of the GRID experiments. There were done two separate experiments using open fields with nominal doses of 2, 5, and 10 Gy, four cell flasks per dose. However, from the film dosimetry in the flask, we know that the dose received in the flasks when irradiating with 5 Gy in an open field actually is around 4.59 ± 0.55 Gy. This is thus assumed to also be the dose received by the cells given nominally 5 Gy.

Since the dose delivered is linear with exposure time, the true dose received when irradiating with 2 Gy and 10 Gy can be found simply by multiplying the result from 5 Gy with 2/5 and 2 respectively. From this we have: 2 Gy → 1.84 Gy, 5 Gy → 4.59 Gy, 10 Gy → 9.18 Gy, which are the actual doses delivered to and responded by the cells, and thus the doses used in fitting the LQ-model. A 95% confidence band to the LQ-curve was also obtained and saved as a look-up table, based on the uncertainty of the survival data and not including the uncertainty of the dose points used in the modelling. The result is shown in Figure 4.35, and uncertainties and p-values of the parameters found are listed in Table 4.10. The survival fraction (SF) was found to be:

$$SF(D) = e^{-\alpha D - \beta D^2} = e^{-0.074 \cdot D - 0.026 \cdot D^2} \tag{4.8}$$

Table 4.10: Uncertainties and p-values of α and β estimated in the LQ-model.

	value	σ	p-value
α	0.074	0.027	$1.2e^{-2}$
β	0.026	0.003	$7.3e^{-7}$

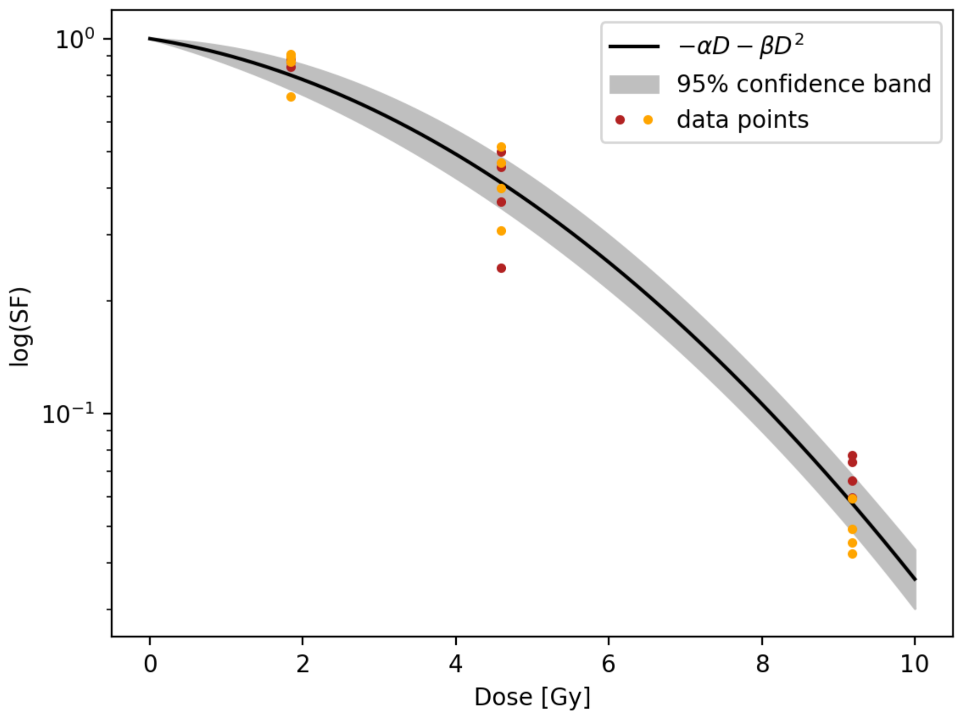


Figure 4.35: LQ-curve fitted to survival from open field irradiation with X-rays with 95% confidence band. Survival data from the same experiments are marked with the same color.

4.4.2 Prediction of cell survival and observed cell survival for X-rays

Combining the LQ-model with the known dose distribution in the cell flasks when using GRID gives the expected cell survival for GRID-irradiation. Again, the GRID irradiations of cells were done using nominal doses of 2, 5 and 10 Gy, and the acquired dose distribution – which is only found for a nominal dose of 5 Gy – is multiplied with 2/5 and 2 to get the desired dose distributions. As the observed cell survival is counted using bands of 0.5 and 1.5 mm, the same is done with the dose profiles to get appropriate comparisons. The uncertainty of the predicted cell survival is a combination of the uncertainty in the dose profiles and from the LQ-model, where confidence intervals for dose profiles of 2 and 10 Gy were found similarly to the original, using equation (4.6) with $\sigma_{D_{fit}}$ being found from the look-up table from the spline interpolation. The observed cell survival is found using an average of all cell flasks receiving the same nominal dose with GRID, in all 8 cell flasks per dose. The uncertainty here is a normal 95% CI based on the standard deviation of the observations and the number of them. The relative percentage difference (RPE) between the predicted and observed survival was found from:

$$\text{RPE}\ (\%) = 100 \cdot \frac{|\,\text{SF}_{\text{obs}} - \text{SF}_{\text{pred}}\,|}{(\text{SF}_{\text{obs}} + \text{SF}_{\text{pred}})/2} \qquad (4.9)$$

Figure 4.36 - Figure 4.41 shows the predicted vs observed cell survival. With a nominal dose of 2 Gy there is no significant difference for the predicted and observed survival. With a nominal dose of 5 Gy there is no significant difference either, but there is a tendency of the observed survival to be somewhat lower at the peak dose areas, with an RPE of around 30%. For a nominal dose of 10 Gy however, there is a significantly lower observed survival at the peak dose areas. The RPE is here around 160 % when using a band width of 0.5 mm and around 140% when using a band width of 1.5 mm.

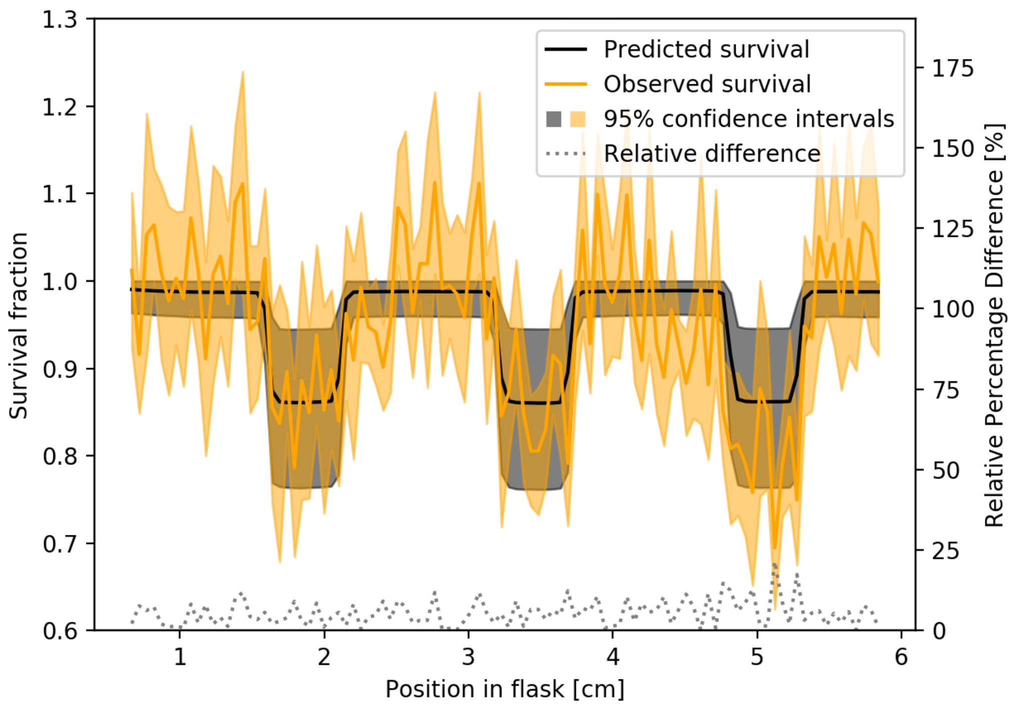


Figure 4.36: Predicted and observed cell survival after X-ray GRID irradiation with a nominal dose of 2 Gy counted by dividing the flask into bands of Δx = 0.5 mm.

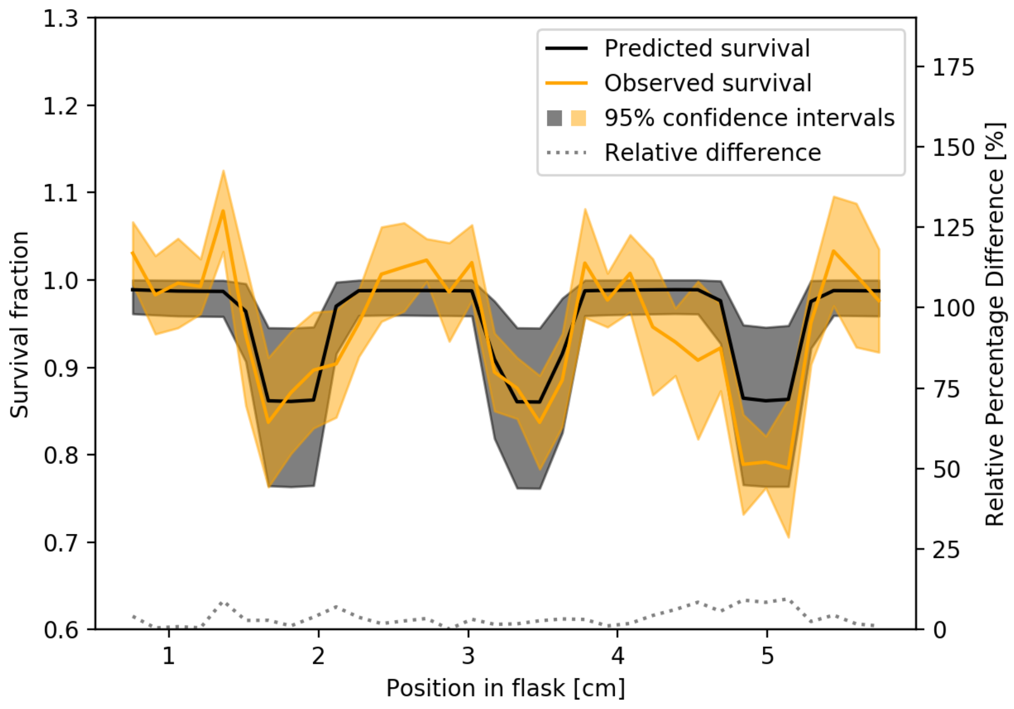


Figure 4.37: Predicted and observed cell survival after X-ray GRID irradiation with a nominal dose of 2 Gy counted by dividing the flask into bands of Δx = 1.5 mm.

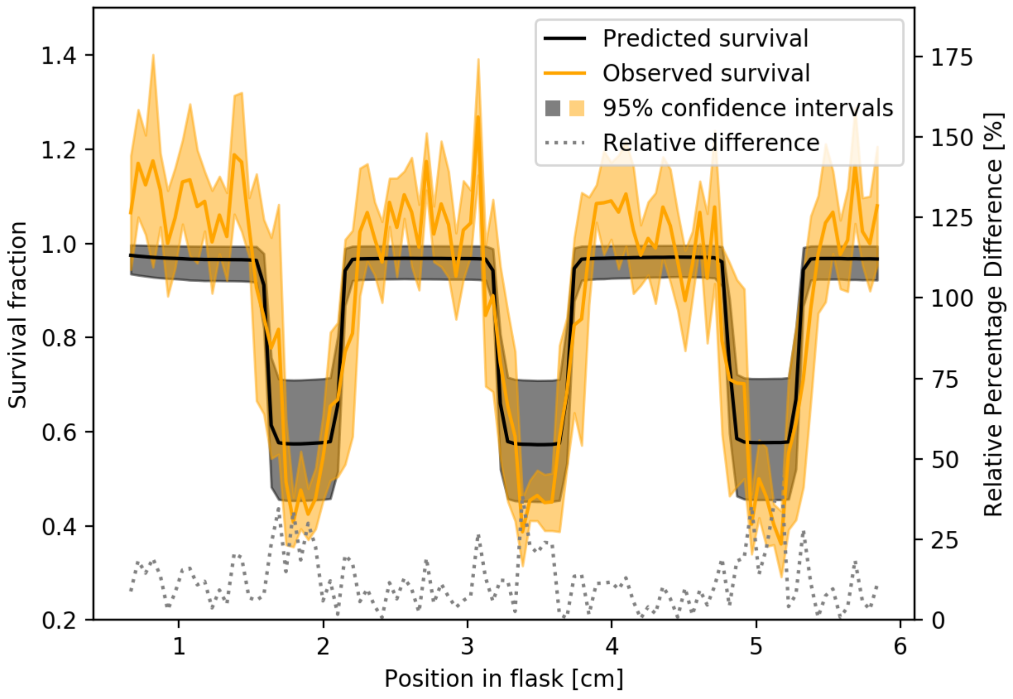


Figure 4.39: Predicted and observed cell survival after X-ray GRID irradiation with a nominal dose of 5 Gy counted by dividing the flask into bands of Δx = 0.5 mm.

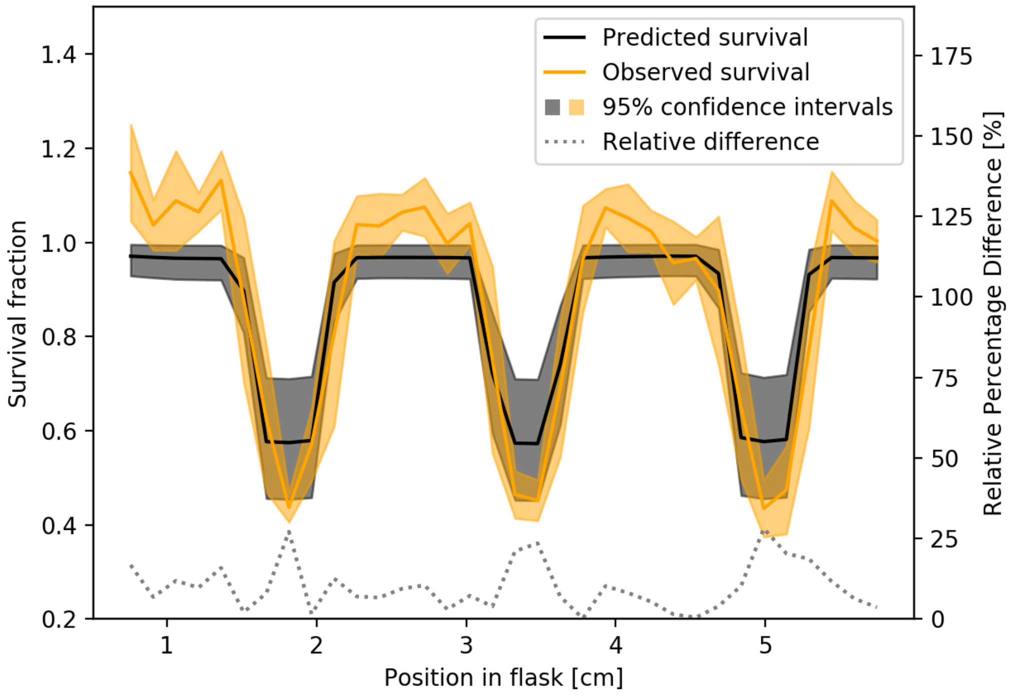


Figure 4.38: Predicted and observed cell survival after X-ray GRID irradiation with a nominal dose of 5 Gy counted by dividing the flask into bands of Δx = 1.5 mm.

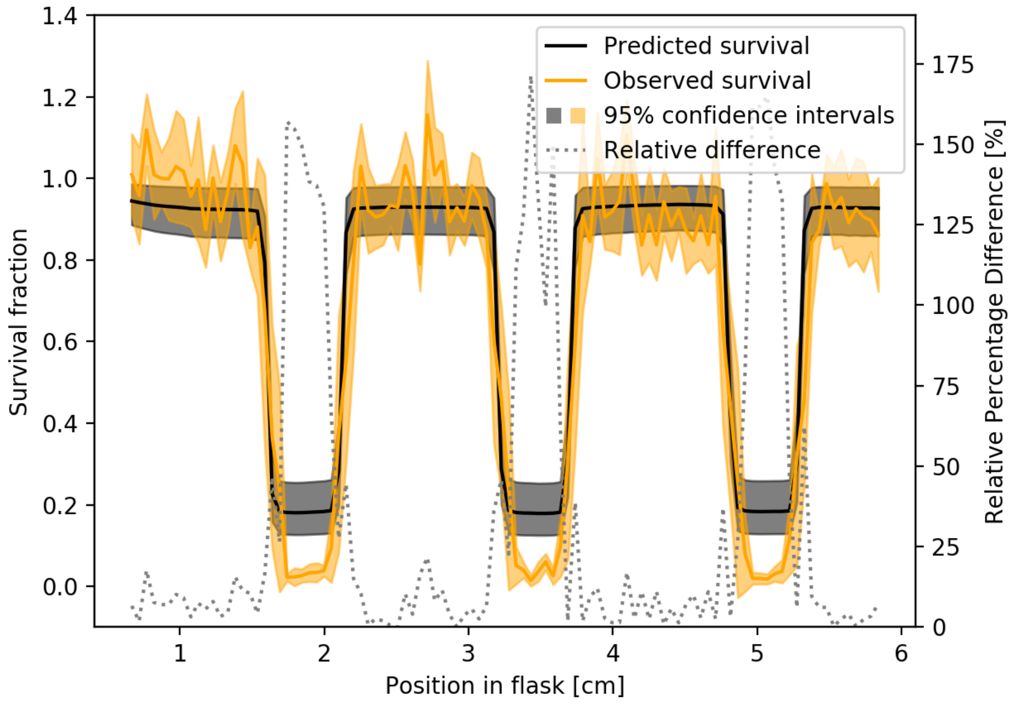


Figure 4.40: Predicted and observed cell survival after X-ray GRID irradiation with a nominal dose of 10 Gy counted by dividing the flask into bands of Δx = 0.5 mm.

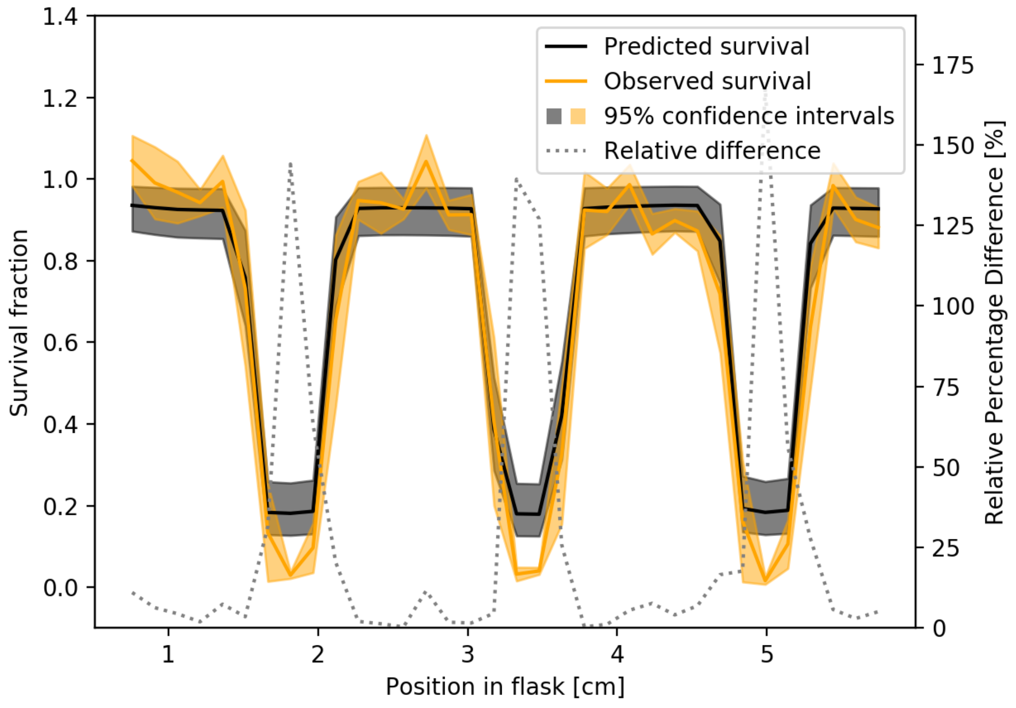


Figure 4.41: Predicted and observed cell survival after X-ray GRID irradiation with a nominal dose of 10 Gy counted by dividing the flask into bands of Δx = 1.5 mm.

4.4.3 Prediction of cell survival and observed cell survival for protons.

Examples of cell irradiated is shown in Figure 4.42. Drawing profiles through the cell dishes would give false profiles if not aligned horizontally. Some of the dishes was therefore rotated with a phase correlation later if it was deemed necessary, using the proton GRID films as a template. However, since the cells are counted within bands, some slack in the alignment was tolerated.

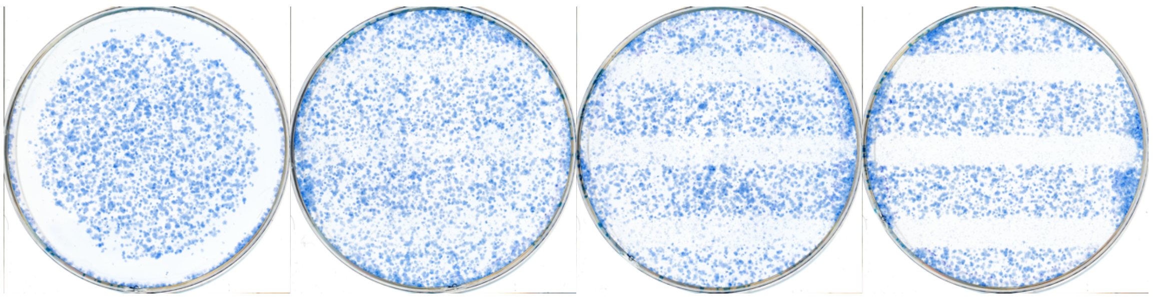


Figure 4.42: Cells irradiated with proton GRID. From left to right: Unirradiated control, nominal doses of 2, 5 and 10 Gy. The unirradiated cell dish was swept with a cotton swab creating the circle with no cells.

In order to do an inspection of observed versus predicted cell survival for proton GRID irradiation, the LQ-model established for X-ray irradiation was used. The predicted and observed survival as well as their uncertainties were found completely equivalent as they were for X-rays. However, since the proton GRID films had a resolution of 150 dpi while the cell scans had 1200 dpi, the actual band width used when extracting the profiles differs slightly. After the profiles were obtained, the cell profile was interpolated down to match the dose profile as the latter was minimally smaller, and the profiles were matched through a phase correlation. There is still some inconsistency in the match between the two, but this is purely due to the acquisition of the profiles as explained.

The observed versus predicted survival for proton irradiation is shown in Figure 4.43 - Figure 4.47 This time, there is no significant difference in the peak dose area for any of the doses. There is however a higher observed survival in one of the valley dose areas, consistently for all doses and most visible when counting using a band width of 1.5 mm. The RPE was found here too, however, the most prominent peaks in the RPE are due to the still poorly matched profiles.

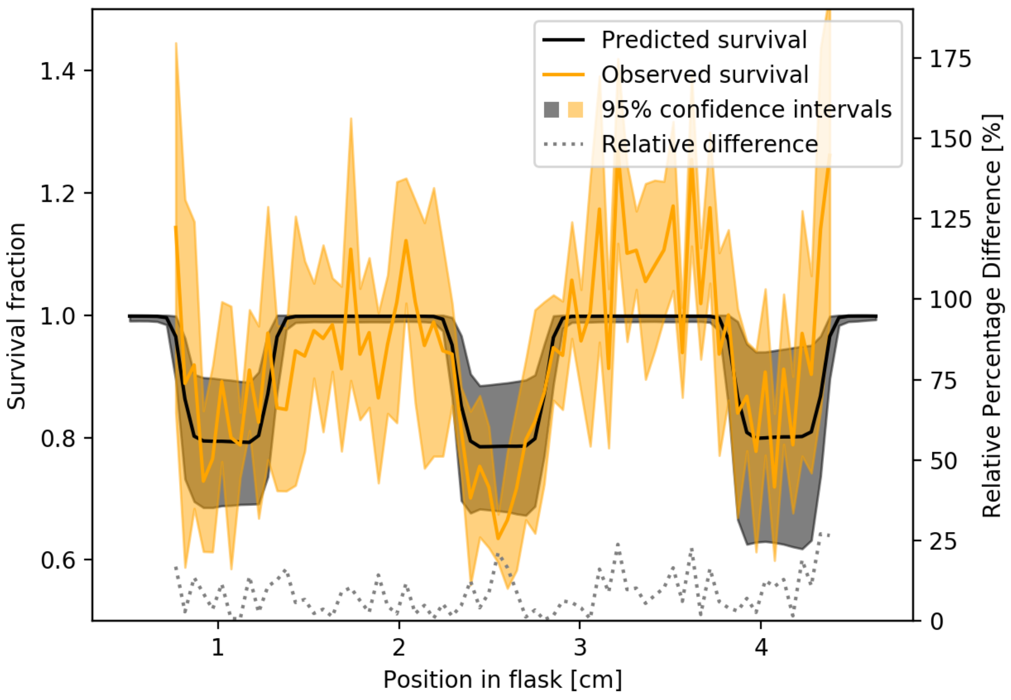


Figure 4.43: Predicted and observed cell survival after proton GRID irradiation with a nominal dose of 2 Gy counted by dividing the flask into bands of Δx = 0.5 mm.

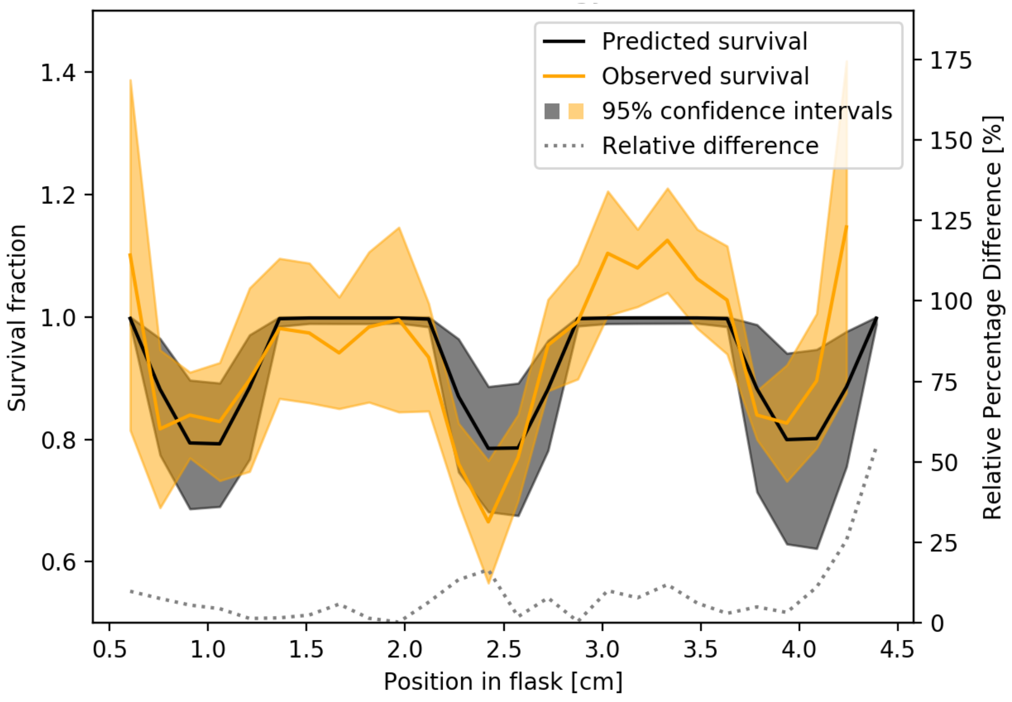


Figure 4.44: Predicted and observed cell survival after proton GRID irradiation with a nominal dose of 2 Gy counted by dividing the flask into bands of Δx = 1.5 mm.

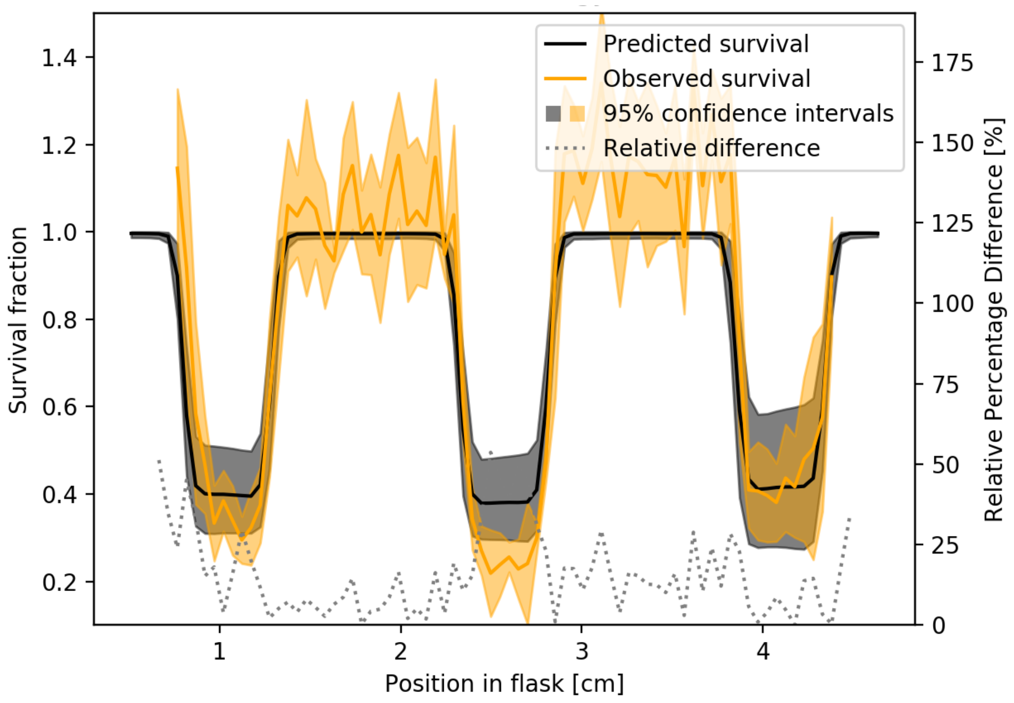


Figure 4.46: Predicted and observed cell survival after proton GRID irradiation with a nominal dose of 5 Gy counted by dividing the flask into bands of Δx = 0.5 mm.

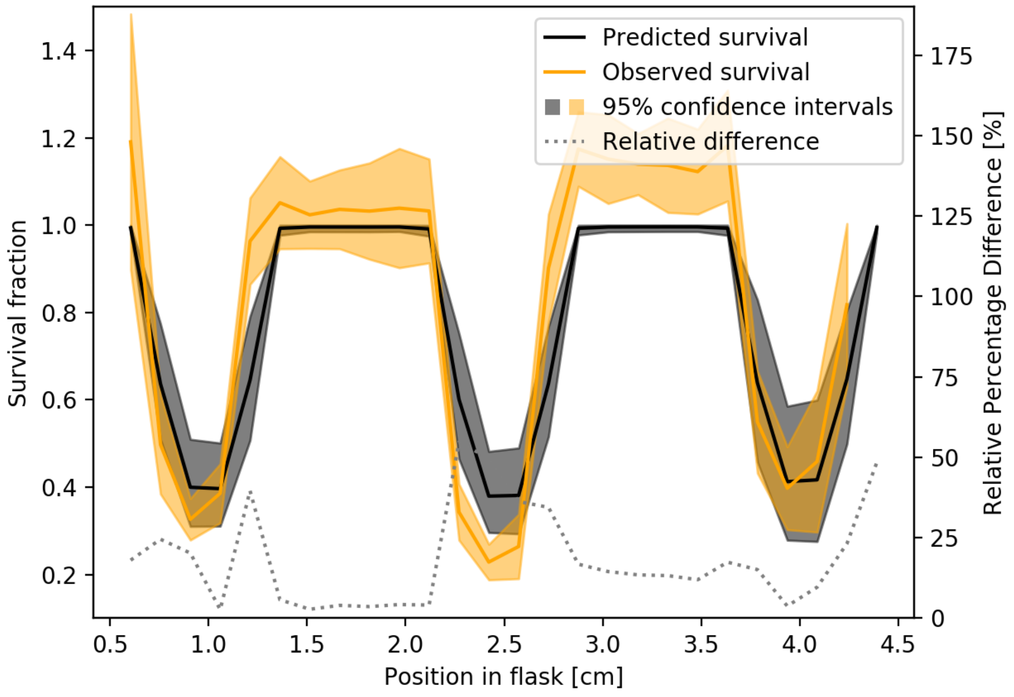


Figure 4.45: Predicted and observed cell survival after proton GRID irradiation with a nominal dose of 5 Gy counted by dividing the flask into bands of Δx = 1.5 mm.

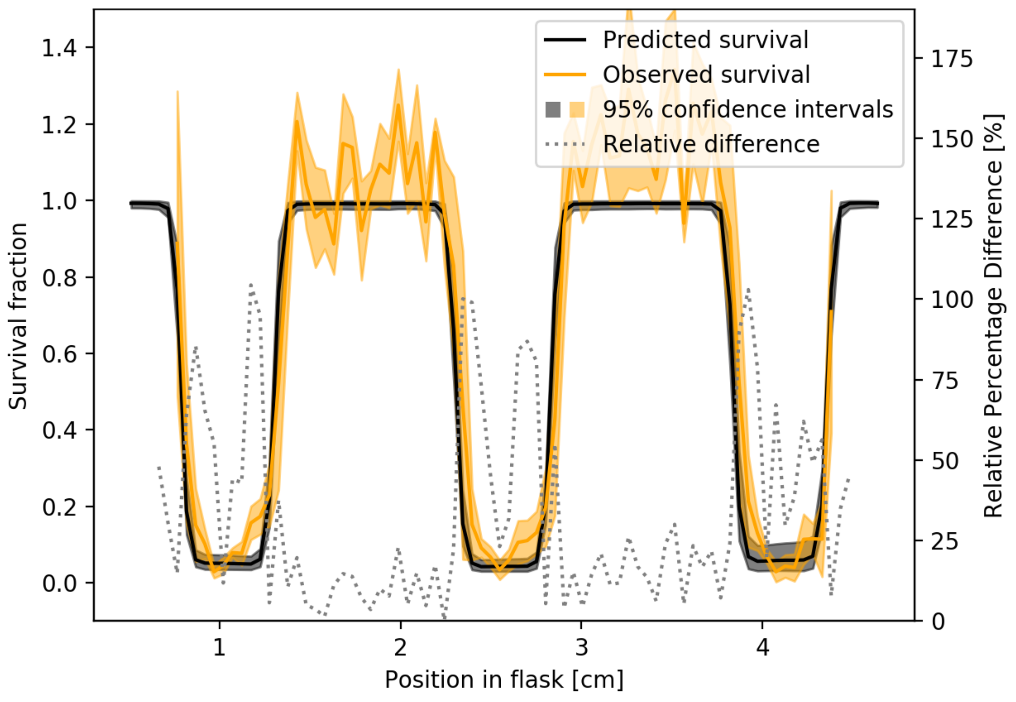


Figure 4.48: Predicted and observed cell survival after proton GRID irradiation with a nominal dose of 10 Gy counted by dividing the flask into bands of Δx = 0.5 mm.

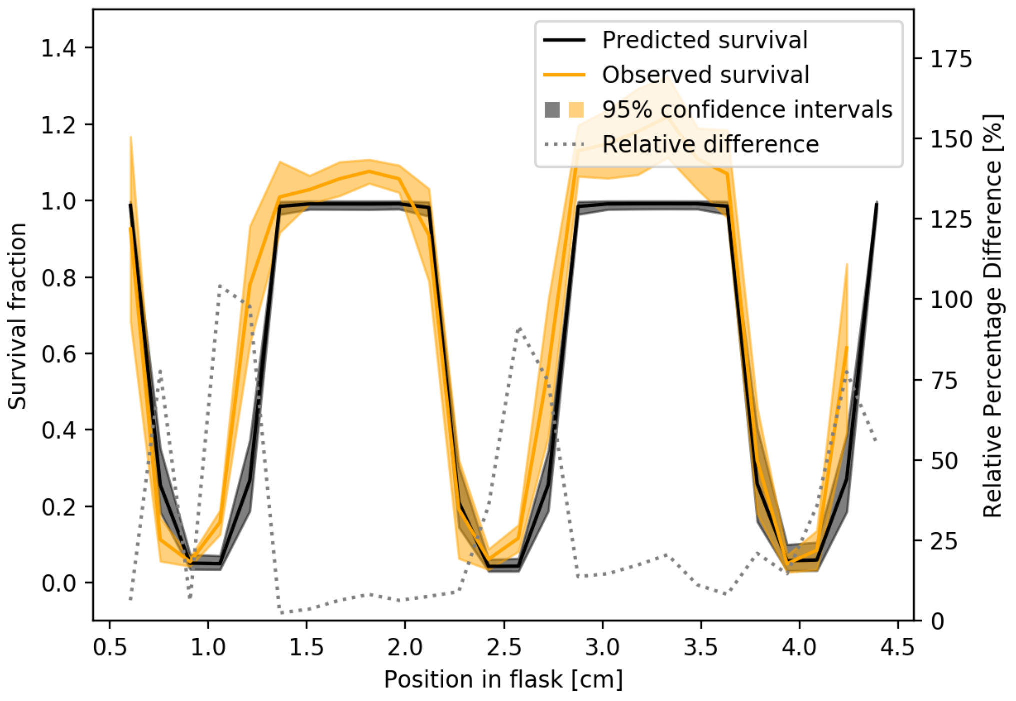


Figure 4.47: Predicted and observed cell survival after proton GRID irradiation with a nominal dose of 10 Gy counted by dividing the flask into bands of Δx = 0.5 mm.

4.4.4 *Average dose deposited and average cell survival*

A comparison of the response with GRID and open fields can be done by looking at the average survival within the cell flasks, and whether the survival seen in conjunction with the average dose delivered with GRID matches the LQ-curve based on the open field irradiations. The average dose received on EBT3 films within the cell flask for the different radiation setups is shown in Table 4.11, illustrating the difference of the total dose delivered with GRID vs open fields. Figure 4.49 shows the average overall survival for both GRID and open fields.

Table 4.11: Average overall dose received

Nominal dose	2 Gy	5 Gy	10 Gy
X-ray, open field	1.84 Gy	4.57 Gy	9.17 Gy
Proton, open field	1.90 Gy	4.76 Gy	9.52 Gy
X-ray, GRID	0.49 Gy	1.23 Gy	2.47 Gy
Proton, GRID	0.66 Gy	1.67 Gy	3.34 Gy

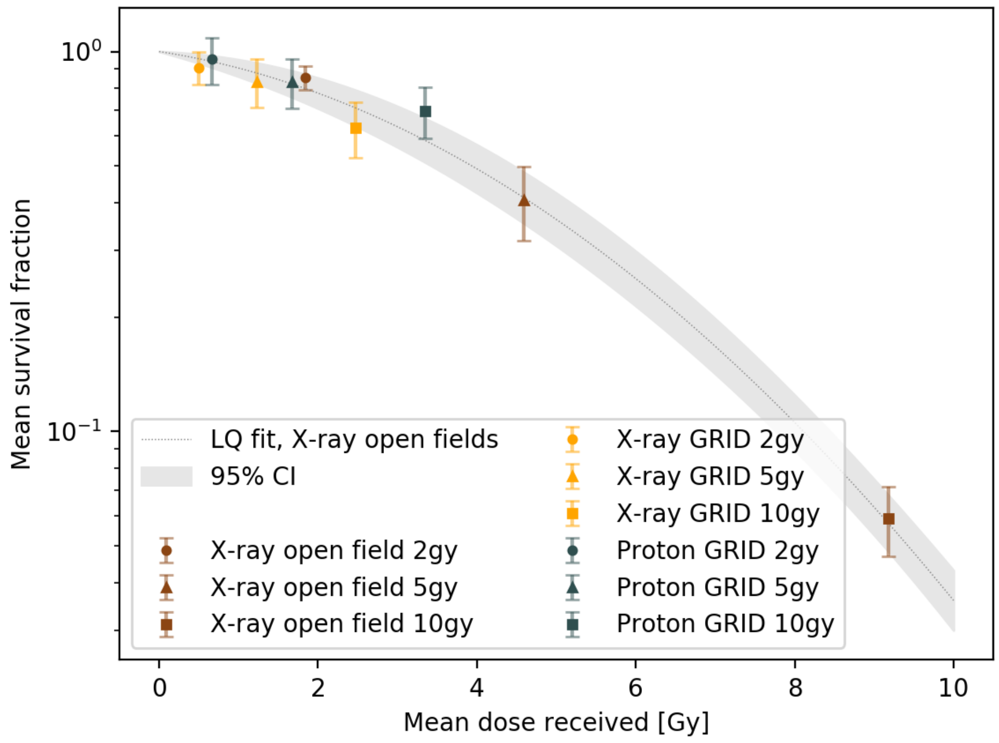


Figure 4.49: Average survival of cells against the average dose deposited in cell flask for the different irradiation modalities.

5 Discussion

5.1 Methodology

In this thesis a methodology for studying in vitro effects of GRID irradiation has been established. This has been done by developing a procedure for Gafchromic EBT3 film dosimetry with GRID irradiation, as well as a means for comparison of the dose distribution to cell survival when cells undergo the same irradiation conditions.

First, the dosimetry of the X-ray chamber was established, both in terms of homogeneity and dose rate, as well as the dose rate inside of T25 flasks positioned in a PMMA cell flask holder. Gafchromic EBT3 films were irradiated with a selection of doses spanning the doses planned given to cells, with extra dose points toward the lower end of the scale, as well as unirradiated films. Films were subsequently irradiated inside of T25 flasks with GRID and an open field. The films were scanned, the calibration films had their netOD extracted from a centered ROI, and an appropriate model converting netOD to dose was chosen after thorough analyses. This was then was applied to all films irradiated in T25 flasks, and the average dose profiles through the flasks were found.

Having survival data provided from cell experiments, a program was developed to count colonies in the cell flasks, divided into bands of thickness Δx. Cell survival from open field irradiations were found, as well as the actual doses delivered to cells irradiated in open fields from the open field film irradiation inside T25 flasks. The combination of these doses and cell survival was used to acquire an LQ-model giving expected cell survival as a function of dose. Combining this LQ-model with the dose deposition obtained from the Gafchromic EBT3 GRID film dosimetry gave an expected cell survival as function of the position in the T25 flasks. This was then compared with the actual observed cell survival from GRID irradiation for different doses.

5.2 X-rays

Regarding the cell counting, every colony was only is counted once with their size being ignored. As larger colonies include more living cells, and colonies might contribute in several bands, this does not give a complete picture of the cell response but rather only the colony survival. All cell analyzes was done dividing the flasks into bands of both thickness 0.5 and 1.5

mm. Although Δx = 1.5 mm gave considerably less noise, the profiles are better preserved using Δx = 0.5 mm which also provides more data from peak dose areas, which are areas of particular interest. Using both Δx gives a comprehensive overview, and the choice of Δx in the future should depend on the type of analyses wanted.

Looking at the open field X-ray irradiation inside T25 flasks with a nominal dose of 5 Gy, the response was 4.59 ± 0.64 Gy on average. One reason for the lower dose is due to the way ion chamber dosimetry was performed inside the T25 flasks. The insertion of the ion chamber into the flask as shown in Figure 3.3 led to a diagonal positioning of the chamber. The sensitive area of the ion chamber was therefore not in the same height as the active layer of the EBT3 films and the chamber thus received slightly higher doses. The dose received will also have differed for different parts of the chamber due to the diagonal positioning, possibly influencing readout. An approximation of the true dose delivered to the flask floor can be calculated using the inverse-square law, seen in equation (5.1), using the height of the middle part of the sensitive area of the chamber, which was elevated around 0.9 cm above the T25 flask floor. The distance from the X-ray source to the Perspex plate was 60 cm, the height of the PMMA holder was 1.5 cm, and the T25 flask floor laid 0.22 cm over the ground. The films irradiated inside the flask calculated to be given 5 Gy were actually given:

$$\frac{\text{intensity}_1}{\text{intensity}_2} = \frac{\text{distance}_1^2}{\text{distance}_2^2} \tag{5.1}$$

$$\text{Dose at flask floor} = \frac{\left(60 - (1.5 + 0.22 + 0.9)\right)^2}{\left(60 - (1.5 + 0.22)\right)^2}\ 5\ \text{Gy} = 4.85\ \text{Gy} \tag{5.2}$$

This can help explain why films irradiated in open field within the T25 flasks showed a response lower than 5 Gy. Also, when irradiating with GRID, it is expected to get somewhat lower peak doses than the dose in an open field, as some of the dose stemming from scatter will be blocked. Scatter will also contribute and give some dose to areas shielded by the GRID collimator, which is why the valley dose not is zero. The great reduction in the average dose delivered for GRID versus when irradiating in an open field, seen in Table 4.11, is expected as the GRID collimator consisted of openings of 5 mm with 10 mm shielding in between, meaning most of the radiation was blocked resulting in a low average dose.

Looking at the average cell survival of the X-ray GRID irradiation from Figure 4.49, they are all towards the lower part of the survival expected from the LQ fit, although there are no significant difference. More details are found in the figures of predicted and observed cell survival for X-ray GRID irradiation, where the response for nominal doses of 2 and 5 Gy match well with no significant difference. For a nominal dose of 10 Gy however, the observed survival at the peak dose areas is significantly lower than what is predicted. There are several possible explanations for these results, as discussed in the following.

5.2.1 Bystander effects and limitations of LQ-model

The impact of the bystander effect in a spatially modulated field is well established, with literature describing both positive and negative bystander effects in play and suggesting a variety of signal mechanisms (Peng et al. 2017). As spatially fractionated fields like GRID might lead to enhanced survival in low dose areas, reduced survival in high dose areas, or both, an LQ model derived from open field irradiations might not be representative for the survival in a spatially modulated field, as indicated by several studies (Bromley et al. 2009; Butterworth et al. 2011; Suchowerska et al. 2005). The various bystander effects originating with GRID depends on both the delivered dose, often with more effect presenting at higher doses, the cell line, the configuration of the spatially modulated field and the dimensions of it (e.g. striped fields versus dotted fields as well as different sizes/thickness of the stripes or dots etc.). In the present study, it may be that bystander effect first were significantly induced at a nominal dose of 10 Gy, with mainly cytotoxic effects generated leading to a reduced survival at the peak dose areas compared to what the LQ model predicted. However, Peng et al. found that a 5 mm spatially fractionated field (as we also employed) gave no difference in the survival compared to an open field. Thus, no bystander effects were found in that study for that specific GRID pattern, although this admittedly was done with different cell lines and higher energy X-rays.

5.2.2 Errors in LQ modelling

A weakness of the LQ-model used here is that it is fitted to observations found at only 3 different dose levels. Ideally, open field irradiation should be done using several doses as this would give a more accurate and reliable model parameters. The few dose points, particularly for the lower doses is probably the reason why α statistically is not as good as β when inspecting their p-values, since α describes the initial slope of the curve while our first data point is at a nominal dose of 2 Gy. In addition, the goodness of the colony detection by the cell counter program is not verified for high doses, i.e. when the staining of the colony segments is more occult. This is evident from Figure 4.31, where the program seems to have chosen a few cells only in the shadow area or along the edges for the open field flask. This problem arose for both for nominal doses of 5 and 10 Gy with open field irradiation and presents a possibly large uncertainty in the LQ model if the few data points used are unreliable. Further evaluation of the cell counter program when used on open fields should be done.

The uncertainty of the doses actually used to make the LQ-model was not included in the modelling. These uncertainties were in fact rather high, i.e. the dose found from a nominal field of 5 Gy was 4.59 ± 0.64. This should have been included into the LQ-model which then would have a significantly larger uncertainty itself, leading to greater uncertainties in the predicted cell survival. This could possibly have influenced the results and make the predicted and observed survival overlap in the high dose areas with 10 Gy nominal dose.

5.2.3 Errors in dosimetry

There is an inconsistency in the pixel values from both GRID and open field irradiation as well as the calibration series from the two different experiments. Both the difference between the open field and GRID profiles seen in Figure 4.15, and the fact that all films irradiated with GRID shows higher pixel values from experiment 1, while all open field irradiations including the calibration series shows higher pixel values from experiment 2. This indicates an erroneous dosimetry, and possible reasons for this are:

Expired films

The greatest source of uncertainty is the EBT3 films used, as they all expired November 2017 which unfortunately was not discovered until too late. This makes the films considerably less reliable.

Forgotten Nylon6 slab

When conducting experiment 1, the Nylon6 slab was mistakenly not placed over the GRID films. This affects the build-up of the electrons prior to reaching the active layer of the film, and therefore also the dose deposition. A test was subsequently performed at experiment 2, where a film was irradiated with Nylon6 covering half. The dose profile is shown in Figure 5.1, where Nylon6 was placed over the right half. The part without Nylon6 shows a decrease in dose around 0.02%. Even though the dose is not greatly affected, it might be one of the reasons for the greater intra-film variation seen in the GRID films from experiment 1 compared to experiment 2. Without the Nylon6, electrons will have reached different levels of build-up prior to reaching the active layer of the film, leading to greater variation in the dose deposition between the films. This in turn gives a higher uncertainty of the peak dose areas for the GRID films.

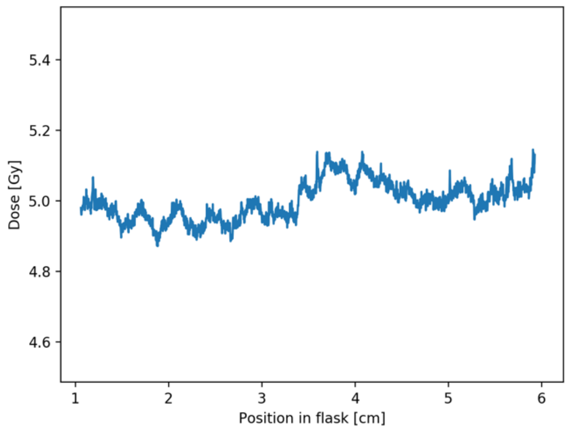


Figure 5.1: Dose profile from a EBT3 film irradiated during experiment 2 with Nylon6 covering the lower the lower (right in figure) half. There is a distinct although small increase in dose in the area where Nylon6 was placed.

Scanning

The scan protocol was not fully developed before the start of this study, resulting in a revision of the protocol during the study and new scans of the films. This was not problematic in itself as it applied to all films, and any influence from the scanner light on the films would be similar for all. However, films from experiment 1 had 67 days between the first and second scan, while films from experiment 2 had 29. Gafchromic films continues to develop with time, and this implies not only a contribution from the scanner light, but also a change in the radiation-induced signal. As the films were expired, it is probable that this development not was consistent for the films and they may have progressed at a different pace. A good indication of this is the difference in the experimental uncertainty of the two calibrations, which has almost doubled for the films from the second scan (Figure 4.8 and Figure 4.20).

Another element is the problems arising from scanning the X-ray films inside of T25 flasks. After being cut and handled, the films do not lie completely flat on the bottom of the flasks, which is the reason why it is recommended to adhere the films to the scanner bed using a glass plate or similar. This was not possible inside of a T25 flask, and the films were inevitably lying at different heights on the scanner bed. Also, the flask used for scanning was cut in two halves, with the top only loosely placed on. This was impossible to do identically for each scan, especially with the weight of the scanner lid pushing and moving the flask top. This will have led to small changes of the height between the scanner lid and bed for the different films. Both causes changes in the pixel intensity. A better solution for the future is to make a template of the outline of the flasks which the films than can be scanned inside of while adhered to the scanner bed.

Calculating netOD

A recommendation found in the literature when calculating netOD is to use each film as their own control, i.e. scanning all films before irradiation as well as after, either in addition to or instead of using films that remain unirradiated the entire time (Devic, Tomic, and Lewis 2016; Devic 2011). Doing this, only the difference in the netOD of the films before and after irradiation would have been measured, and if the difference in the film reading comes from e.g. UV-light exposure or other aspects happening before our experiments started, this method would have handled it. Still, this relies on a completely stable scanner performance, which was not tested in the current study. Optimally, the calibration protocol should have been determined before any irradiation started.

Calibration

The unexplained variation between experiment 1 and 2 lead to the choice of doing one common calibration using both calibration series together. This was deemed the most robust choice, especially since it was rather unclear exactly why the discrepancies occurred. Doing one common calibration would also lead to the difference of the GRID-films moving toward each other when calculating the dose, contrary to doing separate calibration for each experiment or choosing the best series to make one common calibration as both would amplify the difference of the GRID films. This did however lead to negative netOD for some films at the lowest doses, as a weighed mean of the intensity from unirradiated films was subtracted in the calculation of netOD. Using both experiments led to the mean background signal being higher than the signal of some low dose films from experiment 1. To overcome this, the negative netODs were manually set to zero. Still, the calibration in itself and the model selected is very robust and reliable.

5.3 Protons

The average survival for cells irradiated with proton GRID match the predicted LQ-model for 2 and 5 Gy nominal doses perfectly, while 10 Gy nominal dose shows a somewhat although not significantly higher survival. There is also a slightly higher survival than expected present in one of the GRID valley dose areas. Since this only materialize in one of the valley areas it is most likely due to some artefact, and not any biological response like a bystander effect as this would be expected to show in all valley areas. There are some other discrepancies as well, but they can be attributed to the way data is obtained with different resolution of the dose and cell flask scans, leading to slightly different band widths and not perfect matches. There are however some uncertainties in the performance of the proton experiments and analyses worth discussing. Most of the uncertainty regarding the EBT3 films are of course applicable to the films used for proton irradiation, apart from the scanning as this was done with a separate protocol.

5.3.1 Beam uncertainties

The two main sources of uncertainty from the proton experiments are the beam fluctuations and the gradient of the beam. Fluctuations are more of a problem regarding cell experiments, both because this takes more time and fluctuation in the beam thus becomes more probable, and also since by analyzing the films the exact dose given is known. For cell experiments it might have led to small changes in the dose delivery over time. The beam gradient proposes a greater challenge, as this led to the large gradient in the dose profiles (remember that these scans were performed with a glass plate adhering the films to the scan plate, and also the lid fully closed, and the gradient in the profiles are thus purely from the gradient in the proton beam). By chance, the profiles from the different experiments are almost opposite, making the mean profile rather flat, but the uncertainty is predominantly coming from large variations in the beam gradient.

This means that for cell experiments, the dose profile at individual days should be measured due to these large variations. Also, while the orientation of the films and GRID was carefully considered and tracked, cells were irradiated in complete random orientations, which was not marked in any way. This implies that cells irradiated in the same beam may have received different dose distributions, and this will not be similar to the dose distribution found from the film dosimetry. The fact that cell irradiations were done on separate days from the film irradiation adds to this. Still, the high similarity between predicted and observed GRID survival pattern indicates that the proton dosimetry was performed adequately. The inhomogeneous dose distribution might be a reason for the slightly increased survival seen in one of the valley dose areas. Films and cells should preferably be the irradiated in the same orientation, and if achievable also the same day to ensure the dose deposited is as similar as possible.

5.3.2 Cell clustering

Another possible reason for the heightened cell survival in the valley area is the cells tendency to cluster together, particularly at the edges. This is clearly visible in both Figure 4.28, Figure 4.27, and Figure 4.42, as it is an artefact occurring for X-ray cells as well. However, inspecting some images indicates that the problem was larger for protons, and as all colonies in the clusters are counted by the cell counter program, this gives a higher survival than what can be considered "true". Since this problem manifests to a greater degree for the protons, and as the cell dishes used with protons are smaller than for X-rays, it will impact the proton survival more. Also, clustering of the control samples will evidently even the artefact out somewhat, but for half of the proton control samples the edges were wiped with a cotton swab removing any clustered cells.

5.3.3 LQ-modelling

The prediction of cell survival for proton irradiation was done using an LQ-model based on X-ray irradiations. As the biological response to the two different radiation types might differ, the response seen from X-rays are not necessarily true for protons, and optimally a separate LQ-model should have been made based on open field irradiation with protons. However, the relative biological effect (RBE) of photons and protons are 1.0 and 1.1 respectively, and although studies indicate higher biological effects of protons, this is solely toward the more distal end of the Bragg peak where the LET increases (Lühr et al. 2018; Paganetti 2018). Our experiments were performed in front of the Bragg peak, cf. Figure 3.11. The similar RBE for protons and X-rays indicate that similar doses with the two different radiation types should result in similar survival, and the LQ-model based on X-rays thus give a good indication of the survival for protons as well.

5.3.4 Inaccuracies in proton dosimetry

A parafilm sheet was used to get to the desired depth in the Bragg peak. However, the beam also had to penetrate the first polymer layer of the EBT3 film before reaching the active layer, which affects the stopping of the beam and should optimally have been taken into consideration when calculating the MU required. The dose actually delivered is thus higher as the additional polymer layer moves us further back into the Bragg peak. On the opposite side, the correction factor for temperature and pressure, k_{TP}, which neither was included when calculating MUs laid around 1.009 for both experiments, meaning the delivered dose was lower than if the correction factor had been included. However, k_{TP} was not included when calculating the dose to cells either, and most importantly the uncertainty of turning on and off the beam which was done manually by monitoring the transmission chamber, was greater than the variation coming from excluding the polymer layer and k_{TP}.

5.4 Comparison of X-ray GRID and proton GRID

When comparing the dose profiles from X-ray and proton irradiation, Figure 4.25, there is a distinct difference in both peak and valley dose, with proton irradiation seemingly giving significantly higher peak doses as well as significantly lower valley doses. This cannot be fully explained by the errors already discussed, although some differences would be acceptable and even expected. Also, if only looking at the cell survival from both X-ray and proton irradiation, they are virtually perfect matches. This is unexpected if the doses given to cells are truly different for the two radiation types, as suggested by the dosimetry. Since the RBE of the two radiation types are similar, the difference in doses given should manifest as different survival for the two. Knowing this, and also seeing that the predicted survival based on the proton dosimetry is a strong match to the observed survival for all doses used, it is reason to believe that substantially larger errors were associated with the X-ray experiment. This is reinforced by looking at all uncertainties from the X-ray dosimetry and the results discussed there, particularly the results in Figure 4.15. However, despite evidence pointing towards incorrect

X-ray dosimetry, the Monte Carlo simulation and X-ray dosimetry are quite consistent. If MC is regarded to provide the true dose deposition pattern, this points toward good reliability of the X-ray dosimetry, as the MC was performed with an extremely accurate implementation of the set-up. Although MC has a slightly higher peak to valley ratio, this discrepancy is not enough to explain the difference in the dose profiles for X-ray and proton, nor the poor match to the observed survival for 10 Gy X-ray GRID irradiation.

All in all, the results are inconclusive. The X-ray film dosimetry should optimally be redone following the methodology established and taking all suggestions of improvement into account.

5.5 Recommendations for future work

Approaches to improve the experiments conducted here have been suggested above. Most of this work was done compressing 2-dimensional data into 1-dimensional data, only looking at profiles and averages. It would be interesting to develop and perform complete analyses of the 2-dimensional data as well, particularly as this then can be done using additional GRID configurations. Of certain interest would be to use a spotted GRID configuration, as this would resemble a clinical setting. It would also be interesting to elaborate further on striped GRID fields with 2.5 mm thickness, as this is was found to induce bystander effects by Peng et al. MC simulations is a useful tool that should be incorporated when mapping the dosimetry of a GRID set-up, and should be analyzed in a 2-dimensional manner as well.

More cell experiments using GRID irradiation are needed in order to better understand the signaling and response of cells when irradiated with a spatially fractionated field, both in high and low dose areas. Of particular interest would be to perform fractionated experiments using GRID and compare with the same total dose given with one fraction, to see if this has any impact on survival both for peak and valley dose areas. It would also be of interest to do GRID experiments using different types of cell lines, especially to establish any potential difference in response between normal and cancer cell lines. For proton irradiation, both film dosimetry and cell experiments with GRID should be performed at different positions of the Bragg peak, particularly toward the distal end. Since studies have indicated that the increased LET here gives higher RBE at these positions, the potential effect this has on the cell survival under GRID irradiation is an interesting aspect.

After having thoroughly mapped both the dose deposition and cell survival with GRID, the hope would be to combine this and establish a radiobiological model for the cell survival as a function of the dose and thus also the position under the GRID. Learning more about the cell responses during GRID irradiation and establish these models would bring us one step closer to possibly fully implement GRID irradiation as a modality in clinical radiotherapy sometime in the future.

6 References

Alberts., Bruce, Alexander Johnson., Julian Lewis., David Morgan., Martin Raff., Keith Roberts., and Peter Walter. 2015. *Molecular biology of the cell* (Garland Science, Taylor & Francis Group).

Aldelaijan, Saad, and Slobodan Devic. 2018. 'Comparison of dose response functions for EBT3 model GafChromic&#x2122; film dosimetry system', *Physica Medica: European Journal of Medical Physics*, 49: 112-18.

Andreo, P., J. R. Cunningham, K. Hohlfeld, and H. Svensson. 1987. *Absorbed dose determination in photon and electron beams An International Code of Practice* (IAEA: International Atomic Energy Agency (IAEA)).

Andreo., Pedro, David T. Burns., Alan E. Nahum., Jan Seuntjens., and Fran H. Attix. 2017. *Fundamentals of Ionizing Radiation Dosimetry* (Wiley-VCH Verlag).

Attix, Frank Herbert. 1986. *Introduction to Radiologial Physics and Radiation Dosimetry* (WILEY-VCH).

Battistoni, G., J. Bauer, T. T. Boehlen, F. Cerutti, M. P. Chin, R. Dos Santos Augusto, A. Ferrari, P. G. Ortega, W. Kozłowska, G. Magro, A. Mairani, K. Parodi, P. R. Sala, P. Schoofs, T. Tessonnier, and V. Vlachoudis. 2016. 'The FLUKA Code: An Accurate Simulation Tool for Particle Therapy', *Front Oncol*, 6: 116.

Billena, Cole, and Atif J. Khan. 2019. 'A Current Review of Spatial Fractionation: Back to the Future?', *International Journal of Radiation Oncology • Biology • Physics*, 104: 177-87.

Bromley, R., L. Oliver, R. Davey, R. Harvie, and C. Baldock. 2009. 'Predicting the clonogenic survival of A549 cells after modulated x-ray irradiation using the linear quadratic model', *Phys Med Biol*, 54: 187-206.

Burnham., Kenneth P, and David R Anderson. 2002. *Model Selection and Multimodel Inference* (Spriger).

Butterworth, Karl T., Conor K. McGarry, Colman Trainor, Joe M. O'Sullivan, Alan R. Hounsell, and Kevin M. Prise. 2011. 'Out-of-Field Cell Survival Following Exposure to Intensity-Modulated Radiation Fields', *International Journal of Radiation Oncology*Biology*Physics*, 79: 1516-22.

Böhlen, T. T., F. Cerutti, M. P. W. Chin, A. Fassò, A. Ferrari, P. G. Ortega, A. Mairani, P. R. Sala, G. Smirnov, and V. Vlachoudis. 2014. 'The FLUKA Code: Developments and Challenges for High Energy and Medical Applications', *Nuclear Data Sheets*, 120: 211-14.

Castriconi, R., M. Ciocca, A. Mirandola, C. Sini, S. Broggi, M. Schwarz, F. Fracchiolla, M. Martisikova, G. Arico, G. Mettivier, and P. Russo. 2017. 'Dose-response of EBT3 radiochromic films to proton and carbon ion clinical beams', *Phys Med Biol*, 62: 377-93.

Dahle, T. J., A. M. Rykkelid, C. H. Stokkevåg, A. Mairani, A. Görgen, N. J. Edin, E. Rørvik, L. F. Fjæra, E. Malinen, and K. S. Ytre-Hauge. 2017. 'Monte Carlo simulations of a low energy proton beamline for radiobiological experiments', *Acta Oncol*, 56: 779-86.

Dertinger, Hermann, and Horst Jung. 1970. *Molecular Radiation Biology* (Springer-Verlag New York).

Devic, S. 2011. 'Radiochromic film dosimetry: past, present, and future', *Phys Med*, 27: 122-34.

Devic, S., J. Seuntjens, G. Hegyi, E. B. Podgorsak, C. G. Soares, A. S. Kirov, I. Ali, J. F. Williamson, and A. Elizondo. 2004. 'Dosimetric properties of improved GafChromic films for seven different digitizers', *Med Phys*, 31: 2392-401.

Devic, S., N. Tomic, and D. Lewis. 2016. 'Reference radiochromic film dosimetry: Review of technical aspects', *Phys Med*, 32: 541-56.

Fasso., A, A Ferrari., S Roesler., P R Sala, F Ballarini., A Ottolennghi., G Battistoni., F Cerutti., E Gadioli., M V Garzelli., A Empl., and J Ranft. 2003. 'The physics models of FLUKA: status and recent development .', *eConf*, C0303241: MOMT005. 10 p.

Ferrari, Alfredo, Paola Sala, Alberto Fasso, and J. Ranft. 2005. 'FLUKA: a multi-particle transport code', *CERN Yellow report*, 2005-10.

GAFChromic™. 'EBT-3 film spesifications and user guide'. www.gafchromic.com

———. 2016. 'Efficient Protocols for Accurate Radiochromic Film Calibration and Dosimetry'. www.gafchromic.com.

Gholizadeh Sendani, N., A. Karimian, C. Ferreira, and P. Alaei. 2018. 'Technical Note: Impact of region of interest size and location in Gafchromic film dosimetry', *Med Phys*, 45: 2329-36.

Giard, D. J., S. A. Aaronson, G. J. Todaro, P. Arnstein, J. H. Kersey, H. Dosik, and W. P. Parks. 1973. 'In vitro cultivation of human tumors: establishment of cell lines derived from a series of solid tumors', *J Natl Cancer Inst*, 51: 1417-23.

Hall, Eric J, and Amato J Giaccia. 2019. *Radiobiology for the Radiologist* (Wolters Kluwer: Philadelphia).

Hall, Eric J. 2003. 'THE BYSTANDER EFFECT', *Health Physics*, 85: 31-35.

Helstrup, Trygve Holtebekk. Jacob Linder. Håvard. 2019. 'Lineærakselerator'. https://snl.no/lineærakselerator.

Klepp, Olbjørn. 2018. 'Strålebehandling'. https://sml.snl.no/strålebehandling.

Kreftregisteret. https://www.kreftregisteret.no/Registrene/Kreft_i_Norge/.

Kroese., Dirk P, Tim Brereton., Thomas Taimre., and Zdravko I Botev. 2014. 'Why the Monte Carlo method is so important today', *Wiley Interdisciplinary Reviews: Computational Statistics*, 6: 386-92.

Krzempek, D., G. Mianowska, N. Bassler, L. Stolarczyk, R. Kopec, B. Sas-Korczynska, and P. Olko. 2018. 'CALIBRATION OF GAFCHROMIC EBT3 FILM FOR DOSIMETRY OF SCANNING PROTON PENCIL BEAM (PBS)', *Radiat Prot Dosimetry*, 180: 324-28.

Linder, Trygve Holtebekk. Jacob. 2020. 'Akselerator - fysikk'. https://snl.no/akselerator_-_fysikk.

Linder., Trygve Holtebekk. Jacob. 2018. 'Syklotron'. https://snl.no/syklotron.

Liu, Yang, Yinping Dong, Li Kong, Fang Shi, Hui Zhu, and Jinming Yu. 2018. 'Abscopal effect of radiotherapy combined with immune checkpoint inhibitors', *Journal of Hematology & Oncology*, 11: 104.

Lühr, Armin, Cläre von Neubeck, Mechthild Krause, and Esther G. C. Troost. 2018. 'Relative biological effectiveness in proton beam therapy - Current knowledge and future challenges', *Clinical and translational radiation oncology*, 9: 35-41.

Marín, Alicia, Margarita Martín, Olga Liñán, Felipe Alvarenga, Mario López, Laura Fernández, David Büchser, and Laura Cerezo. 2014. 'Bystander effects and radiotherapy', *Reports of practical oncology and radiotherapy : journal of Greatpoland Cancer Center in Poznan and Polish Society of Radiation Oncology*, 20: 12-21.

Mohan, Radhe, and David Grosshans. 2017. 'Proton therapy - Present and future', *Advanced drug delivery reviews*, 109: 26-44.

Ozaki, Toshinori, and Akira Nakagawara. 2011. 'Role of p53 in Cell Death and Human Cancers', *Cancers*, 3: 994-1013.

Paganetti, Harald. 2018. 'Proton Relative Biological Effectiveness - Uncertainties and Opportunities', *International journal of particle therapy*, 5: 2-14.

Peng, Valery, Natalka Suchowerska, Linda Rogers, Elizabeth Claridge Mackonis, Samantha Oakes, and David R. McKenzie. 2017. 'Grid therapy using high definition multileaf collimators: realizing benefits of the bystander effect', *Acta Oncologica*, 56: 1048-59.

Podgorsak, E B. 2005. 'Treatment Machines for External Beam Radiotherapy.' in E B Podgorsak (ed.), *Radiation Oncology Physics: A Handbook for Teachers and Students* (IAEA: Vienna, Austria).

Poh, Sharon S., Melvin L. K. Chua, and Joseph T. S. Wee. 2018. 'Why we should give spatially fractionated radiation therapy (GRID) a second look—especially in nasopharyngeal carcinoma', *Annals of Nasopharynx Cancer*, 2.

Radiology Key, . 2017. https://radiologykey.com/parallel-plate-ionization-chamber/.

Shortt., J P Seuntjens. W Strydom. K R. 2005. 'Dosimetric Principles, Quantities and Units.' in E B Podgorsak (ed.), *Radiation Oncology Physics: A Handbook for Teachers and Students* (IAEA: Vienna, Austria).

Sipilä, Petri, Jarkko Ojala, Sampsa Kaijaluoto, Ilkka Jokelainen, and Antti Kosunen. 2016. 'Gafchromic EBT3 film dosimetry in electron beams - energy dependence and improved film read-out', *Journal of applied clinical medical physics*, 17: 360-73.

Suchowerska, N., M. A. Ebert, M. Zhang, and M. Jackson. 2005. 'In vitro response of tumour cells to non-uniform irradiation', *Phys Med Biol*, 50: 3041-51.

WHO. 'Cancer'. https://www.who.int/news-room/fact-sheets/detail/cancer.

Wikimedia Commons, contributors. 2018. Wikimedia Commons, the free media repository. https://commons.wikimedia.org/w/index.php?title=File:Depth_Dose_Curves.jpg&oldid=285469739.

———. 2019. Wikimedia Commons, the free media repository. https://commons.wikimedia.org/w/index.php?title=File:Linear_accelerator_animation_16frames_1.6sec.gif&oldid=372834616.

Appendix A

Model 1: $a \cdot \text{netOD} + b \cdot \text{netOD}^n$

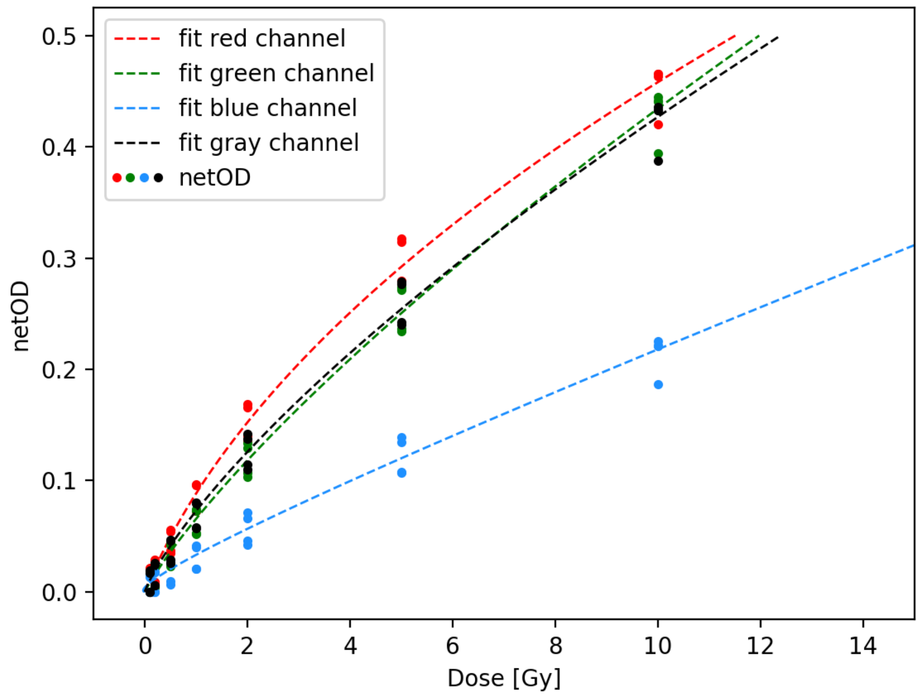


Figure 85: netOD and best fit for all color channels with model 1.

Table 12: Parameters used in model 1 for best fit

	Red	Green	Blue	Gray
n	2.0	1.5	0.75	1.5
a	8.805 ± 1.120	10.308 ± 1.928	72.177 ± 8.084	6.980 ± 1.995
b	28.418 ± 2.831	19.285 ± 3.188	-17.973 ± 5.103	25.160 ± 3.342

$$\sigma_{D_{\text{exp}}}^{\text{model1}} = \sqrt{\left(a + b\,n\,\text{netOD}^{n-1}\right)^2 \sigma_{\text{netOD}}^2}$$

$$\sigma_{D_{\text{fit}}}^{\text{model1}} = \sqrt{\text{netOD}^2\,\sigma_a^2 + \text{netOD}^{2n}\,\sigma_b^2}$$

$$\sigma_{D_{\text{tot}}}^{\text{model1}} = \sqrt{\text{netOD}^2\,\sigma_a^2 + \text{netOD}^{2n}\,\sigma_b^2 + \left(a + b\,n\,\text{netOD}^{n-1}\right)^2 \sigma_{\text{netOD}}^2}$$

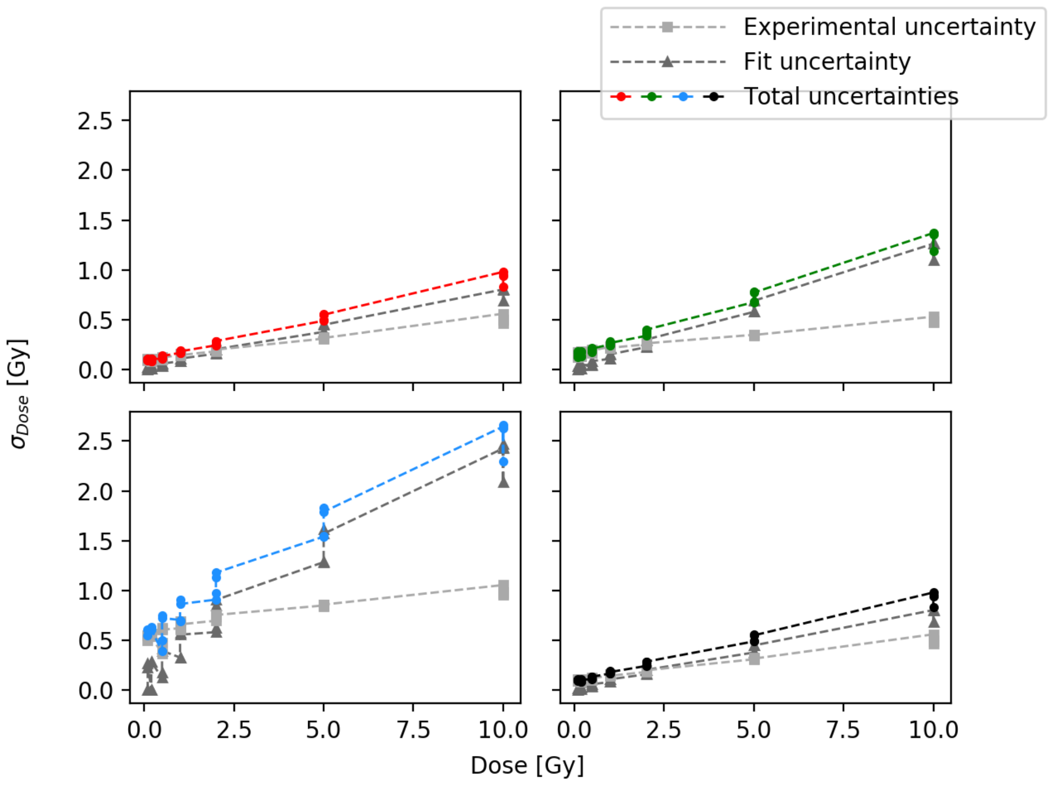


Figure 87: Experimental, fit and total uncertainty of the best fit to each color channel with model 1.

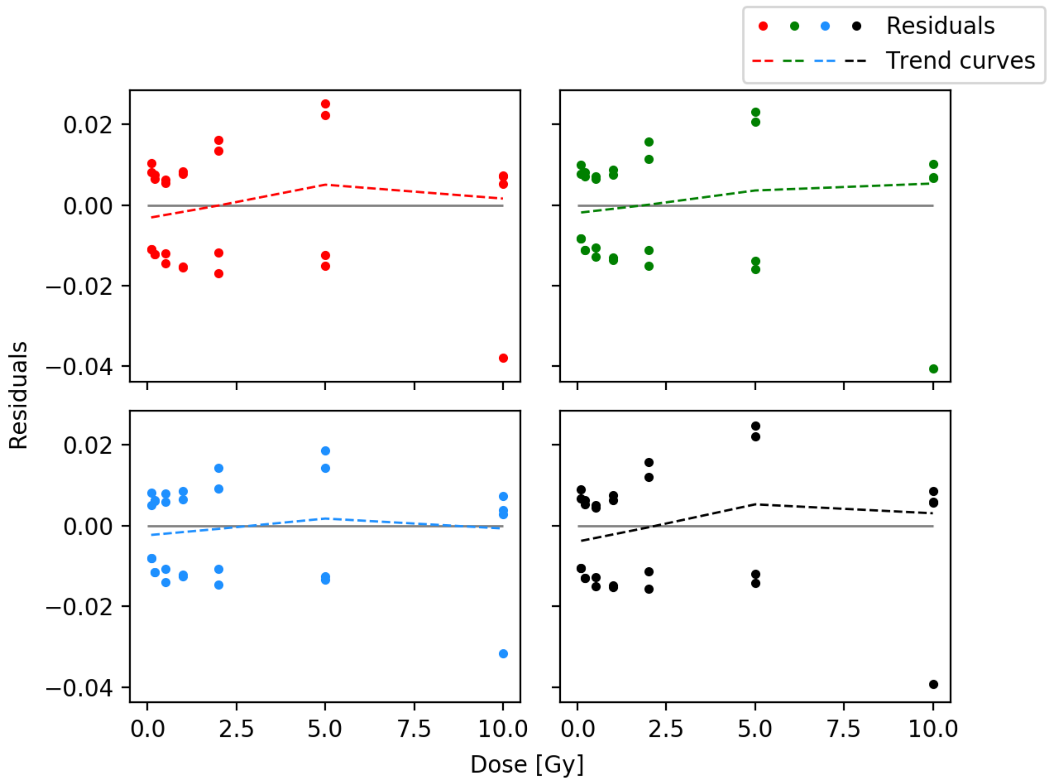


Figure 86: Residuals and loess curves from the best fit to each color channel with model 1.

Model 2: $a + b/(\mathrm{netOD} - c)$

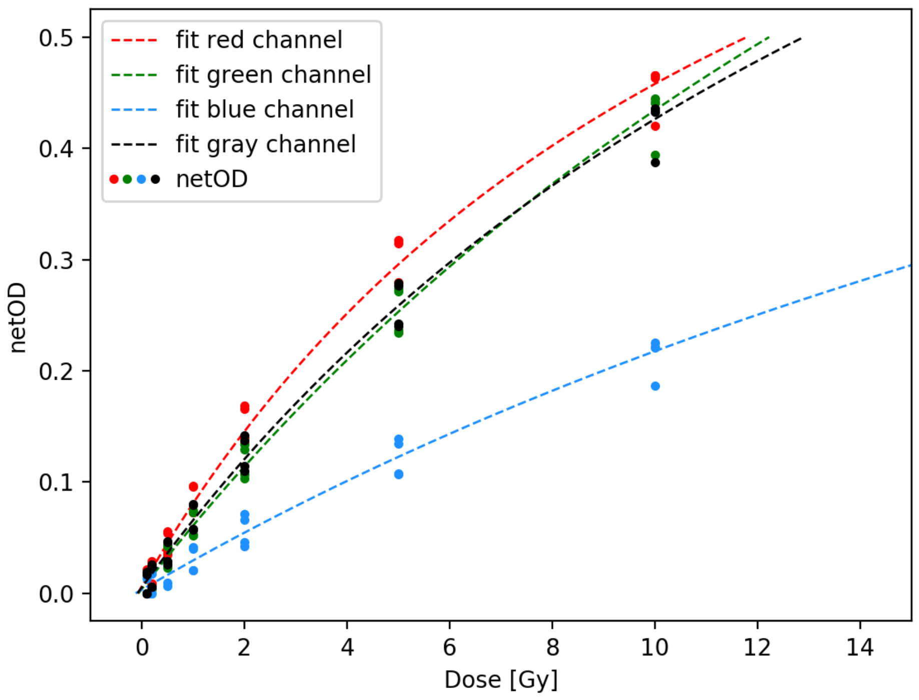


Figure 88: netOD and best fit for all color channels with model 2.

Table 13: Parameters used in model 2 for best fit

	Red	Green	Blue	Gray
a	-12.770 ± 2.405	-26.698 ± 7.636	-38.910 ± 23.064	19.655 ± 4.715
b	-13.138 ± 3.751	-42.173 ± 20.684	-40.814 ± 43.301	-24.546 ± 9.659
c	1.035 ± 0.106	1.584 ± 0.327	1.0522 ± 0.494	1.254 ± 0.197

$$\sigma_{D_{\mathrm{exp}}}^{\mathrm{model2}} = \sigma_{\mathrm{netOD}}\, b\,/(\mathrm{netOD} - c)^2$$

$$\sigma_{D_{\mathrm{fit}}}^{\mathrm{model2}} = \sqrt{\sigma_a^2 + \sigma_b^2/(\mathrm{netOD} - c)^2 + \sigma_c^2\, b^2/(\mathrm{netOD} - c)^4}$$

$$\sigma_{D_{\mathrm{tot}}}^{\mathrm{model2}} = \sqrt{\sigma_a^2 + \sigma_b^2/(\mathrm{netOD} - c)^2 + (\sigma_c^2 + \sigma_{\mathrm{netOD}}^2)\, b^2/(\mathrm{netOD} - c)^4}$$

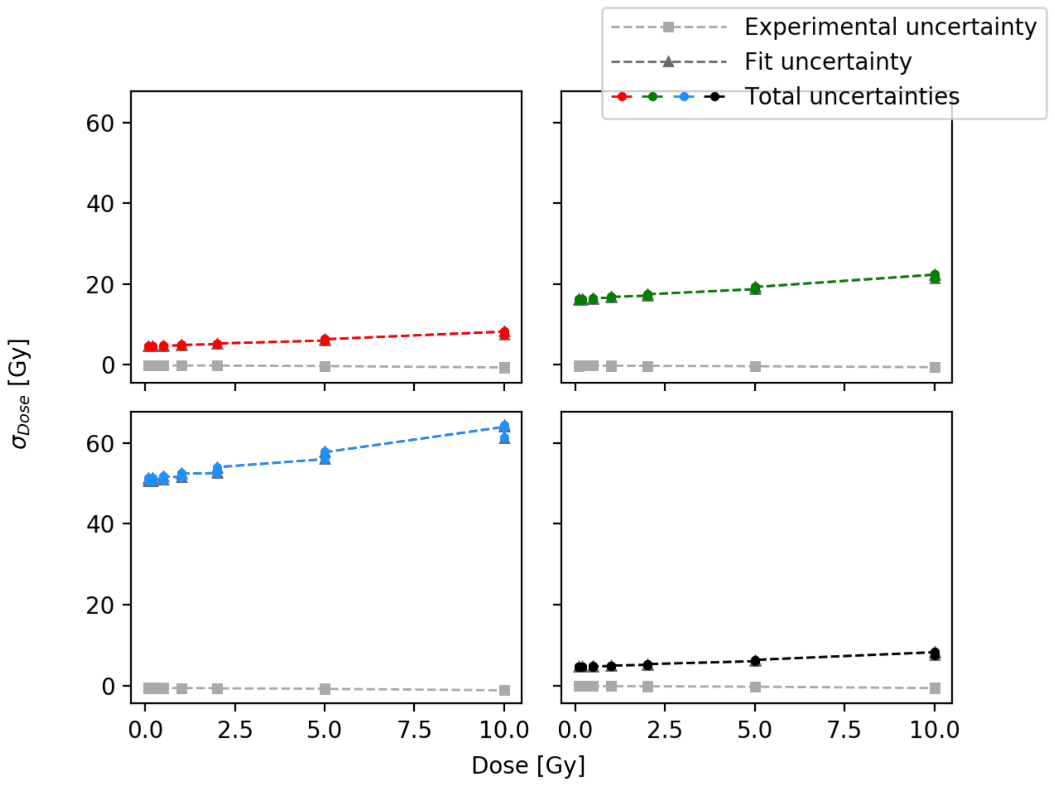


Figure 90: Experimental, fit and total uncertainty of the best fit to each color channel with model 2.

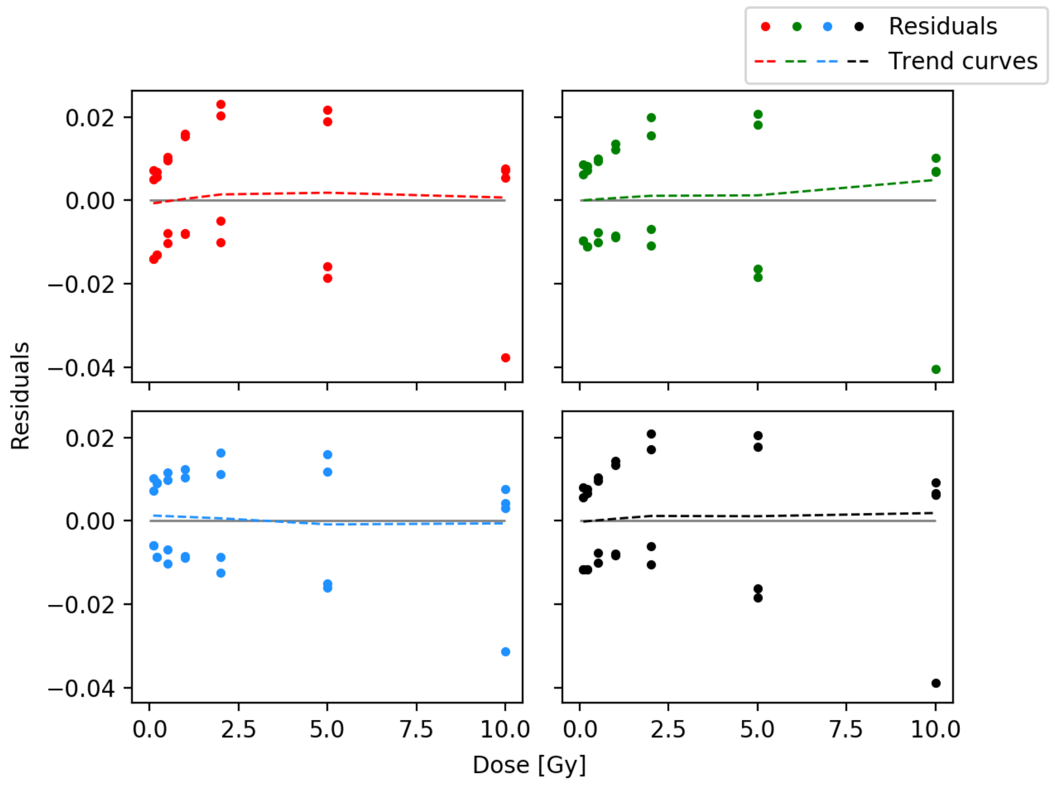


Figure 89: Residuals and loess curves from the best fit to each color channel with model 1.

Model 3: $a \cdot \mathrm{netOD} \cdot \mathrm{e}^{\,b \cdot \mathrm{netOD}}$

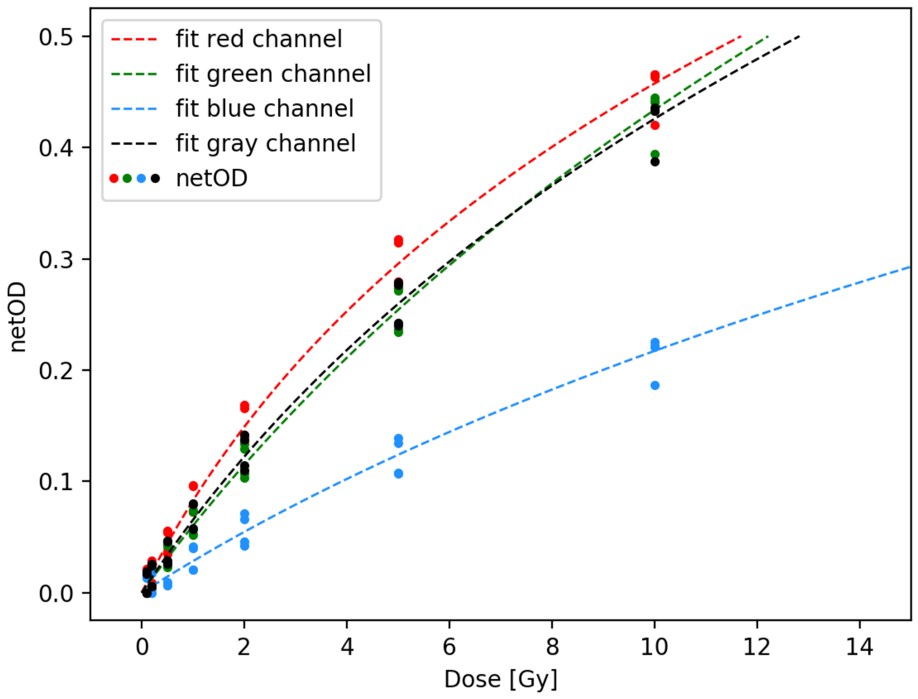


Figure 91: netOD and best fit for all color channels with model 3.

Table 14: Parameters used in model 3 for best fit

	Red	Green	Blue	Gray
a	10.607 ± 0.811	15.731 ± 1.026	33.875 ± 3.348	14.153 ± 0.997
b	1.580 ± 0.182	0.880 ± 0.165	1.410 ± 0.502	1.189 ± 0.181

$$\sigma_{\mathrm{D_{exp}}}^{\mathrm{model3}} = \sigma_{\mathrm{netOD}}\, a\, \mathrm{e}^{b\,\mathrm{netOD}} \sqrt{\left(1 + b\,\sigma_{\mathrm{netOD}} + b^2 \mathrm{netOD}^2\right)}$$

$$\sigma_{\mathrm{D_{fit}}}^{\mathrm{model3}} = \mathrm{e}^{b\,\mathrm{netOD}} \sqrt{\mathrm{netOD}^2\, \sigma_a^2 + a^2\, \mathrm{netOD}^4 \sigma_b^2}$$

$$\sigma_{\mathrm{D_{tot}}}^{\mathrm{model3}} = \mathrm{e}^{b\,\mathrm{netOD}} \sqrt{\mathrm{netOD}^2\, \sigma_a^2 + a^2\, \mathrm{netOD}^4\, \sigma_b^2 + a^2 \left(1 + b\,\sigma_{\mathrm{netOD}} + b^2 \mathrm{netOD}^2\right) \sigma_{\mathrm{netOD}}^2}$$

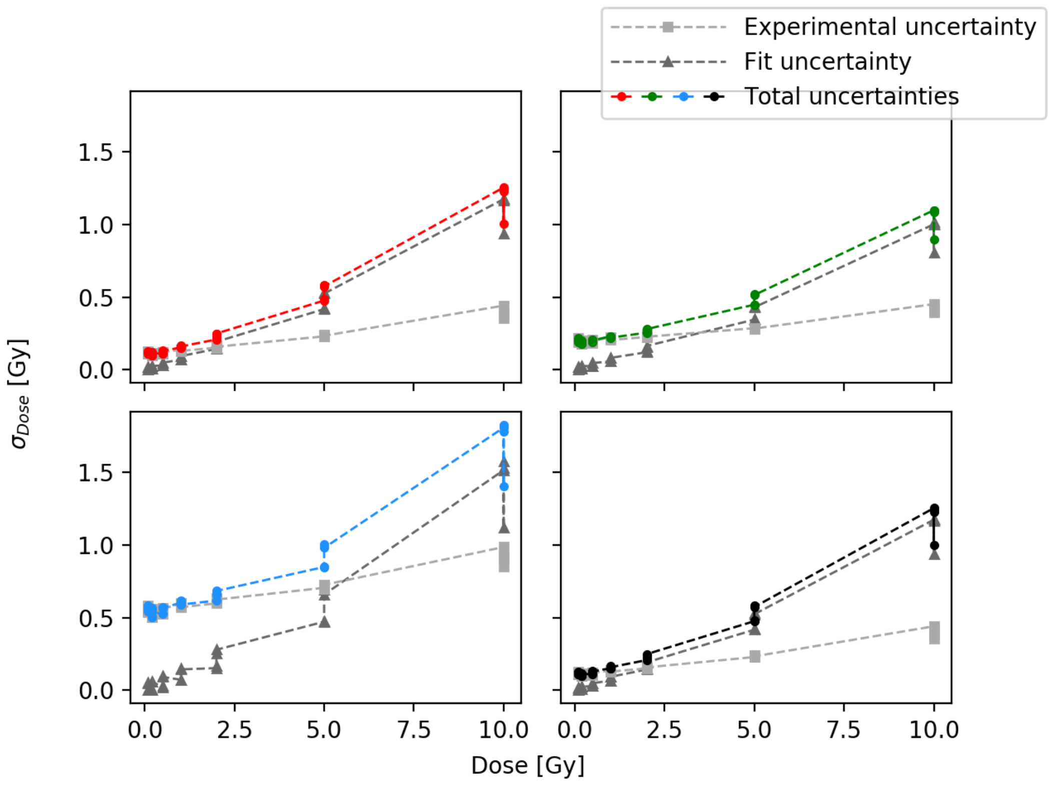


Figure 92: Experimental, fit and total uncertainty of the best fit to each color channel with model 3.

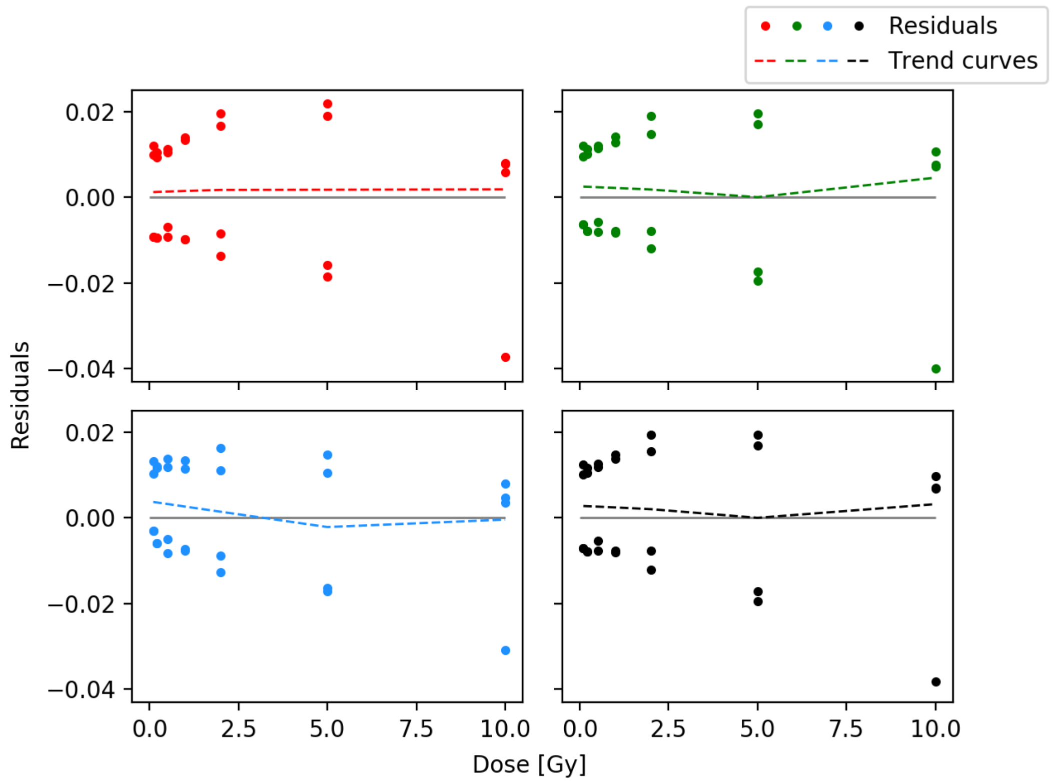


Figure 93: Residuals and loess curves from the best fit to each color channel with model 3.

Model 4: $a \cdot \mathrm{netOD}^n$

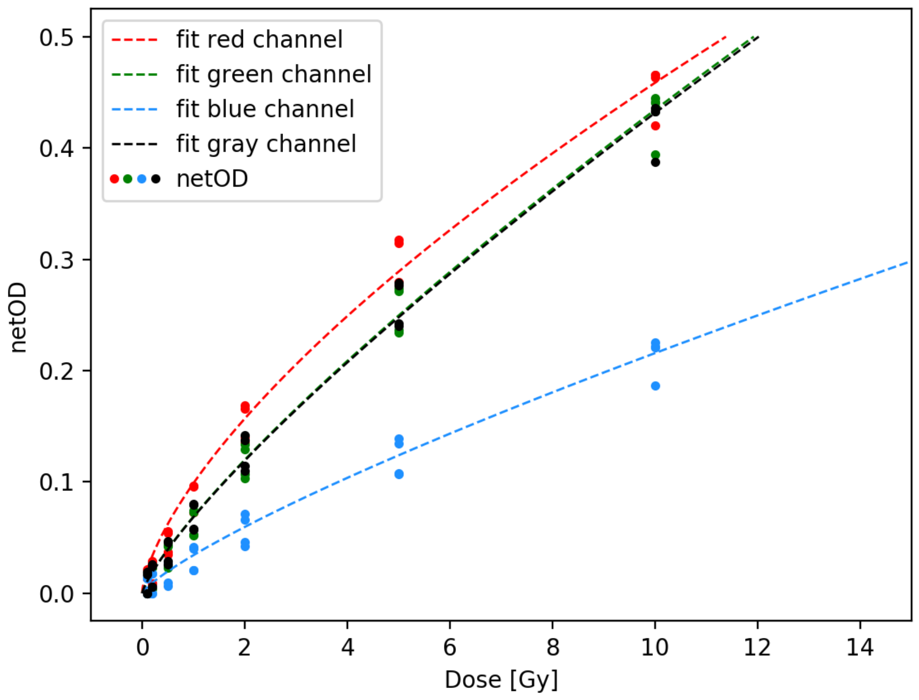


Figure 94: netOD and best fit for all color channels with model 4.

Table 15: Parameters used in model 4 for best fit

	Red	Green	Blue	Gray
n	1.5	1.25	1.25	1.25
a	32.193 ± 0.522	28.358 ± 0.424	68.005 ± 1.627	28.588 ± 0.459

$$\sigma_{\mathrm{D_{exp}}}^{\mathrm{model4}} = \sqrt{\left(a\, n\, \mathrm{netOD}^{n\text{-}1}\right)^2 \sigma_{\mathrm{netOD}}^2}$$

$$\sigma_{\mathrm{D_{fit}}}^{\mathrm{model4}} = \mathrm{netOD}^n\, \sigma_a$$

$$\sigma_{\mathrm{D_{tot}}}^{\mathrm{model4}} = \sqrt{\mathrm{netOD}^{2n}\, \sigma_a^2 + \left(a\, n\, \mathrm{netOD}^{n\text{-}1}\right)^2 \sigma_{\mathrm{netOD}}^2}$$

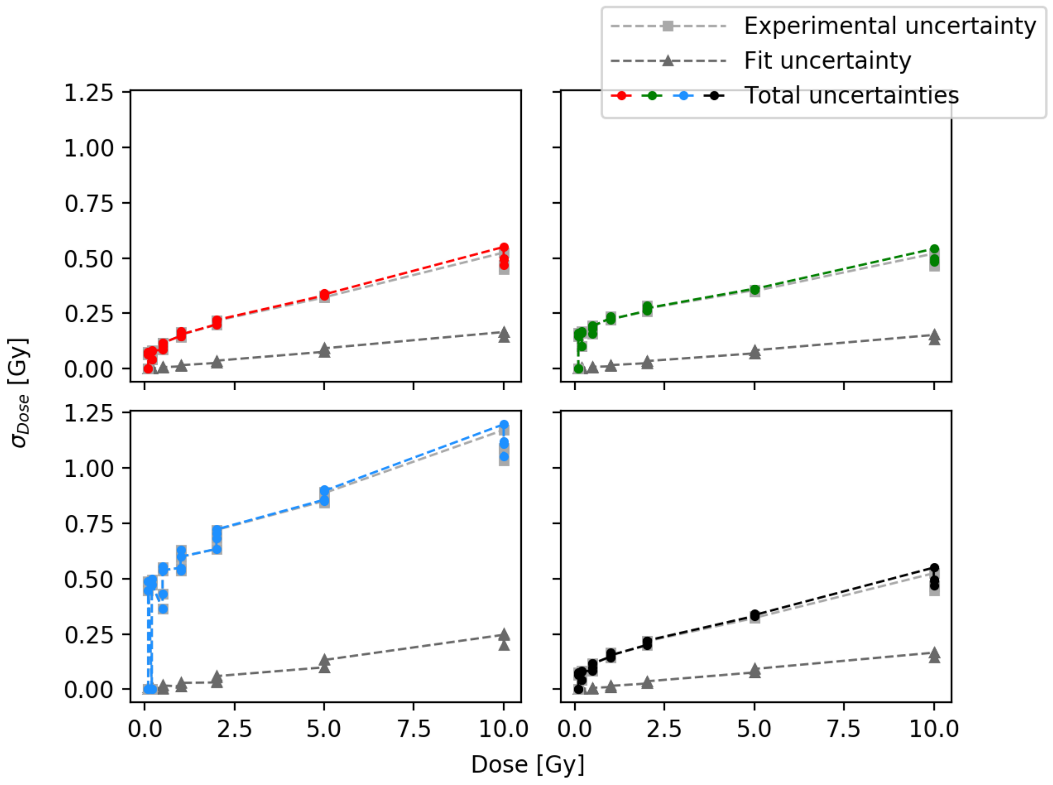


Figure 96: Experimental, fit and total uncertainty of the best fit to each color channel with model 4.

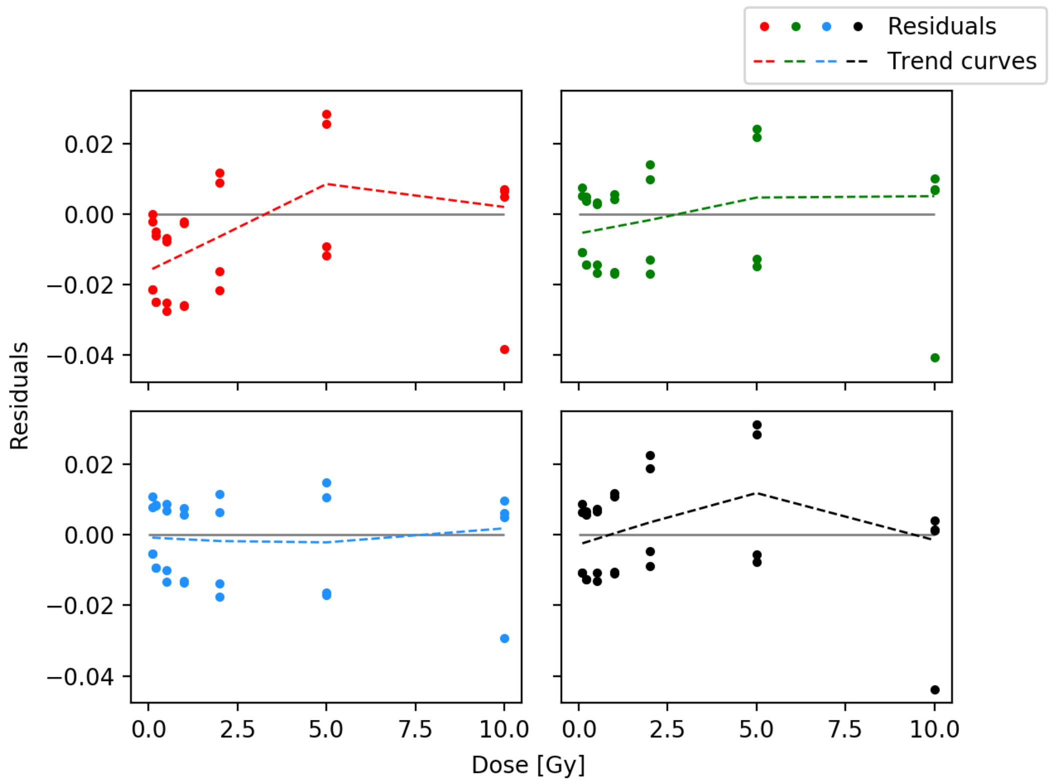


Figure 95: Residuals and loess curves from the best fit to each color channel with model 4.

Appendix B: Code

Chamber dosimetry:

```
#dosimetry of X-ray chamber

import  numpy as np
import matplotlib.pyplot as  plt
from scipy.interpolate import griddata
from scipy import stats
plt.rcParams["lines.linewidth"] = 1
from matplotlib.legend_handler import HandlerTuple

def dose(Mn, T, P, sigma_Mn, factor):
    Nk = 43.77
    sigma_Nk = 0.39
    Pu = 1.02
    mu_rho = 1.075
    kv = 1.
    T0 = 20.
    P0 = 1013.
    korr = (273.2+T)/(273.2+T0)*P0/P
    dose = Mn*Nk*Pu*mu_rho*kv*korr*factor #factor to get it to Gy/min
    error = np.sqrt((dose/Nk)**2 * sigma_Nk**2 + (dose/Mn)**2 * sigma_Mn**2)
    return dose, error


# PERSPEX PLATE

x, y = np.mgrid[0:7:8j, 0:6:7j]
x_fine, y_fine = np.mgrid[0:90:91j, 0:80:81j] #finer grid

point60 =
np.array([[0,0],[0,6],[1,1],[1,3],[1,5],[2,2],[2,4],[3,1],[3,3],[3,5],[4,0],[4,2],[
4,4],[4,6],[5,1],[5,3],[5,5],[6,2],[6,4],[7,1],[7,5]]) #positions of measurements
point60_fine =
np.array([[0,0],[0,60],[10,10],[10,30],[10,50],[20,20],[20,40],[30,10],[30,30],[30,
50],[40,0],[40,20],[40,40],[40,60],[50,10],[50,30],[50,50],[60,20],[60,40],[70,10],
[70,50]])+10

value60rate =
np.array([0.119,0.137,0.159,0.172,0.165,0.176,0.177,0.169,0.182,0.174,0.149,0.178,0
.179,0.164,0.162,0.175,0.167,0.161,0.164,0.136,0.138]) #measured rate
value60charge =
np.array([13.93,16.10,18.64,20.11,19.34,20.64,20.70,19.74,21.34,20.41,17.50,20.73,2
0.94,19.28,19.07,20.62,19.61,18.88,19.09,15.99,16.09]) #measured charge

result_rate60 = griddata(point60_fine, value60rate, (x_fine,y_fine), method =
"cubic") #visualisation with finer grid
```

```
result_charge60 = griddata(point60, value60charge, (x,y), method = "linear")
#calculation remaining points
sigma_Mn = np.std(result_charge60[2:6,2:5]) #[2:6,2:5] points inside of homogenous
field

print np.mean(result_charge60[2:6,2:5])
print np.std(result_charge60[2:6,2:5])
print "charge ", dose(np.mean(result_charge60[2:6,2:5]), 24, 1017, sigma_Mn,
0.001/2)[0]
print "error ", dose(np.mean(result_charge60[2:6,2:5]), 24, 1017, sigma_Mn,
0.001/2)[1]

plt.imshow(result_rate60)
plt.set_cmap("cividis")
plt.xticks(np.arange(10, 80, 10))
plt.yticks(np.arange(10, 90, 10))
plt.grid(color = "k")
plt.show()


# low dose on Perspex plate

x = np.linspace(0,30,100)
timepoints = np.repeat([5,10,15,20], 2)
measurements = np.array([[0.36,0.36],[1.24,1.22],[2.11,2.15],[2.96,3.06]])
lowdose, error = dose(measurements, 27, 1018, 0, 0.001)

slope, intersect, r_value,  p_value, std_err = stats.linregress(timepoints,
np.ravel(lowdose))
print slope, intersect
print std_err

plt.plot(x, x*slope + intersect, "--k")
plt.plot(timepoints, np.ravel(lowdose), "o", color = "orange")
plt.xlabel("Time [sec]")
plt.ylabel("Dose [Gy]")
plt.legend([r"$0.00865\cdot$t - 0.026","Data points" ])
plt.show()


#dosimetry in FLASKS

flaskA = np.array([25.43,25.27]) #measured charges
flaskB = np.array([25.29,25.22])
flaskC = np.array([25.38,25.33])
flaskD = np.array([25.20,25.27])

doseA, errA = dose(np.mean(flaskA), 24, 1017, np.std(flaskA), 0.001/2)
doseB, errB = dose(np.mean(flaskB), 24, 1017, np.std(flaskB), 0.001/2)
doseC, errC = dose(np.mean(flaskC), 24, 1017, np.std(flaskC), 0.001/2)
doseD, errD = dose(np.mean(flaskD), 24, 1017, np.std(flaskD), 0.001/2)
```

```
meanDose = np.mean([doseA, doseB, doseC, doseD])
errDose = np.mean([errA, errB, errC, errD])

print "A: mean %.3f +/- %.3f, dose %.4f +/- %.4f " %(np.mean(flaskA),
np.std(flaskA), doseA, errA)
print "B: mean %.3f +/- %.3f, dose %.4f +/- %.4f " %(np.mean(flaskB),
np.std(flaskB), doseB, errB)
print "C: mean %.3f +/- %.3f, dose %.4f +/- %.4f " %(np.mean(flaskC),
np.std(flaskC), doseC, errC)
print "D: mean %.3f +/- %.3f, dose %.4f +/- %.4f " %(np.mean(flaskD),
np.std(flaskD), doseD, errD)
print "Mean dose %.3f +/- %.4f"  %(meanDose, errDose)

#low dose in flasks

a = np.array([[0.36,0.38],[1.45,1.49],[2.72,2.59],[3.72,3.75]])
b = np.array([[0.35,0.34],[1.47,1.46],[2.80,2.63],[3.72,3.91]])
c = np.array([[0.48,0.34],[1.58,1.41],[2.69,2.68],[3.79,3.67]])
d = np.array([[0.50,0.52],[1.60,1.54],[2.66,2.49],[3.72,3.71]])
mean = np.reshape(np.stack((a,b,c,d), axis = 1),(4,8))

slopeA, intersectA, r_valueA,  p_valueA, std_errA = stats.linregress(timepoints,
dose(np.ravel(a), 27, 1018, 0, 0.001)[0])
slopeB, intersectB, r_valueB,  p_valueB, std_errB = stats.linregress(timepoints,
dose(np.ravel(b), 27, 1018, 0, 0.001)[0])
slopeC, intersectC, r_valueC,  p_valueC, std_errC = stats.linregress(timepoints,
dose(np.ravel(c), 27, 1018, 0, 0.001)[0])
slopeD, intersectD, r_valueD,  p_valueD, std_errD = stats.linregress(timepoints,
dose(np.ravel(d), 27, 1018, 0, 0.001)[0])

print slopeA, intersectA, std_errA
print slopeB, intersectB, std_errB
print slopeC, intersectC, std_errC
print slopeD, intersectD, std_errD
slope, intersect, r_value,  p_value, std_err =
stats.linregress(np.repeat(timepoints,4), dose(np.ravel(mean), 27, 1018, 0,
0.001)[0])
print slope, intersect
print std_err

a_plot, = plt.plot(x, x*slopeA + intersectA,  "--", color = "orange")
b_plot, = plt.plot(x, x*slopeB + intersectB, "--", color = "green")
c_plot, = plt.plot(x, x*slopeC + intersectC, "--",color = "firebrick")
d_plot, = plt.plot(x, x*slopeD + intersectD, "--", color = "deepskyblue")
a_data = plt.plot(timepoints, dose(np.ravel(a), 27, 1018, 0, 0.001)[0], "o", color
= "orange")[0]
b_data = plt.plot(timepoints, dose(np.ravel(b), 27, 1018, 0, 0.001)[0], "o", color
= "green")[0]
c_data = plt.plot(timepoints, dose(np.ravel(c), 27, 1018, 0, 0.001)[0], "o", color
= "firebrick")[0]
```

```
d_data = plt.plot(timepoints, dose(np.ravel(d), 27, 1018, 0, 0.001)[0], "o", color
= "deepskyblue")[0]
mean, = plt.plot(x, x*slope + intersect, "-k")

plt.legend([a_plot,b_plot,c_plot,d_plot, mean,
(a_data,b_data,c_data,d_data)],["Position A","Position B","Position C","Position
D", r"Mean, $0.01093\cdot$t - 0.035", "data points"], handler_map={tuple:
HandlerTuple(ndivide=None)})
plt.xlabel("Time [sec]")
plt.ylabel("Dose [Gy]")
plt.show()
```

Film calibration:

```
#find best model D(netOD) from calibration films

import numpy as np
import matplotlib.pyplot as plt
import cv2
import glob
import os
from scipy.optimize import curve_fit
from statsmodels.nonparametric.smoothers_lowess import lowess
from scipy import interpolate
from scipy import stats
from matplotlib.legend_handler import HandlerTuple
plt.rcParams["lines.linewidth"] = 1.


def find_model1(dose,netOD,color): #find the  best n for each color channel, model1
    N = np.linspace(0,5,21)
    rsq = []
    for n in N:
        def model(netOD,a,b):
            return a*netOD + b*netOD**n
        popt, pcov = curve_fit(model, netOD, dose, method = "lm")
        rsq.append(r_squared_adjusted(dose, model(netOD, *popt), n=28,  p = 
len(popt)))
    print "Best fit ", color," n =", N[np.argmax(rsq)],", r^2 =", np.max(rsq)
    n = N[np.argmax(rsq)]

    def model1(netOD, a,b ):
        return a*netOD + b*netOD**n

    def sigma_Dmodel1(param, netOD, sigma_param, sigma_netOD): #uncertainty of dose 
fit
        a,b, =  param
        sigma_a, sigma_b = sigma_param

        D_tot = np.sqrt(netOD**2 * sigma_a**2 + netOD**(2*n) * sigma_b**2 + (a + 
n*b*netOD**(n-1))**2 * sigma_netOD**2) #total
        D_fit = np.sqrt(netOD**2 * sigma_a**2 + netOD**(2*n) * sigma_b**2)
        D_exp = np.sqrt((a + n*b*netOD**(n-1))**2 * sigma_netOD**2)
        return D_tot, D_fit, D_exp

    return model1, sigma_Dmodel1

def model2(netOD,a,b,c):
    return a + b/(netOD-c)
```

```
def sigma_Dmodel2(param, netOD, sigma_param, sigma_netOD):
    a,b,c = param
    sigma_a, sigma_b, sigma_c = sigma_param
    D_tot = np.sqrt(sigma_a**2 + sigma_b**2/(netOD-c)**2 + (b**2/(netOD-
c)**4)*(sigma_c**2 + sigma_netOD**2))
    D_fit = np.sqrt(sigma_a**2 + sigma_b**2/(netOD-c)**2 + sigma_c**2 *
b**2/(netOD-c)**4 )
    D_exp = b*sigma_netOD/(netOD-c)**2
    return D_tot, D_fit, D_exp

def model3(netOD,a,b):
    return a*netOD*np.exp(b*netOD)

def sigma_Dmodel3(param, netOD, sigma_param, sigma_netOD):
    a,b = param
    sigma_a, sigma_b = sigma_param
    D_tot = np.exp(b*netOD) * np.sqrt(netOD**2*sigma_a**2 +
a**2*netOD**4*sigma_b**2 + a**2*(1+b*sigma_netOD+b**2*netOD**2)*sigma_netOD**2)
    D_fit = np.exp(b*netOD) * np.sqrt(netOD**2*sigma_a**2 +
a**2*netOD**4*sigma_b**2)
    D_exp = np.exp(b*netOD)*sigma_netOD*a*np.sqrt(1 + b*sigma_netOD +
b**2*netOD**2)
    return D_tot, D_fit, D_exp

def find_model4(dose, netOD, color): #find the  best n for each color channel,
model4
    N = np.linspace(0,5,21)
    rsq = []
    for n in N:
        def model(netOD,a):
            return a*netOD**n
        popt, pcov = curve_fit(model, netOD, dose, method = "lm")
        rsq.append( r_squared_adjusted(dose, model(netOD, *popt), n= 28, p =
len(popt)))
    print "Best fit ", color," n =", N[np.argmax(rsq)],", r^2 =", np.max(rsq)
    n = N[np.argmax(rsq)]

    def model4(netOD, a):
        return a*netOD**n

    def sigma_Dmodel4(a, netOD, sigma_a, sigma_netOD):
        D_tot = np.sqrt(netOD**(2*n) * sigma_a**2 + (a*n*netOD**(n-1))**2 *
sigma_netOD**2)
        D_fit = np.sqrt(netOD**(2*n) * sigma_a**2)
        D_exp = np.sqrt((a*n*netOD**(n-1))**2 * sigma_netOD**2)
        return D_tot, D_fit, D_exp

    return model4, sigma_Dmodel4


def set_roi(image):
    min = image.shape[1]/2 - 95  #190 pixels = 4 mm, 142 pixels = 3 mm
    max = image.shape[1]/2 + 95
    return image[min:max,min:max]


def find_netOD(roi, i_unexp, i_bckg, sigma_unexp, sigma_bckg):
    i_exp = np.mean(roi)
    sigma_exp = np.std(roi)
    netOD = np.log10((i_unexp-i_bckg)/(i_exp-i_bckg))
    std = np.sqrt((sigma_unexp + sigma_bckg)/(i_unexp - i_bckg)**2 + (sigma_exp**2
+ sigma_bckg)/(i_exp - i_bckg)**2)/np.log(10.)
    return netOD, std


def i_unexposed(unexposed_films_list): #also used to find i_background, returns a
wieghed mean of the intensity and the uncertainty.
    red  = 0; green = 0; blue = 0; gray = 0
    uncert_red  = 0; uncert_green = 0; uncert_blue = 0; uncert_gray = 0
    for file in  unexposed_films_list:
        uexposed_film = cv2.imread(file, cv2.IMREAD_UNCHANGED)
        gray_img = cv2.cvtColor(uexposed_film, cv2.COLOR_BGR2GRAY)

        red_roi = set_roi(uexposed_film[:,:,2])
        green_roi = set_roi(uexposed_film[:,:,1])
        blue_roi = set_roi(uexposed_film[:,:,0])
        gray_roi = set_roi(gray_img)

        red += np.mean(red_roi)/np.std(red_roi)**2
        uncert_red += 1./np.std(red_roi)**2
        green += np.mean(green_roi)/np.std(green_roi)**2
        uncert_green += 1./np.std(green_roi)**2
        blue += np.mean(blue_roi)/np.std(blue_roi)**2
        uncert_blue += 1./np.std(blue_roi)**2
        gray += np.mean(gray_roi)/np.std(gray_roi)**2
        uncert_gray += 1./np.std(gray_roi)**2
    return red/uncert_red, green/uncert_green, blue/uncert_blue, gray/uncert_gray,
1./uncert_red, 1./uncert_green, 1./uncert_blue, 1./uncert_gray

def r_squared_adjusted(points, fit, n, p): #n = samle size, p = variables in model
    average = np.sum(points)/len(points)
    expl_var = np.sum((fit - points)**2)
    tot_var = np.sum((fit-average)**2)
    r_squared = 1.-(expl_var/tot_var)
    return 1. - (1.-r_squared)*(n-1)/(n-p-1.)

def rss(points, fit): #residual sum of squares
    return np.sum((points-fit)**2)

def aicc(points, fit, k, n): #Corrected akaike criterion, k = number of parameter
to be fitted in the model, n = sample size
    rss = np.sum((points-fit)**2)
    return 2.*k + n*np.log(rss) + (2.*k**2 + 2.*k)/(n-k-1.)
```

```
def run_model(netOD_red, netOD_green, netOD_blue, netOD_gray, model, uncertainty,
std_netOD_red, std_netOD_green, std_netOD_blue, std_netOD_gray, testmodel = True):
    res_red = []; res_green = []; res_blue = []; res_gray = []

    if testmodel == False: #separate for model1 and 4 as n is different for each
color channel
        model_red, sigmaD_red = model(doses,netOD_red,"red")
        model_green, sigmaD_green = model(doses,netOD_green,"green")
        model_blue, sigmaD_blue = model(doses,netOD_blue,"blue")
        model_gray, sigmaD_gray = model(doses,netOD_gray,"gray")

        popt_red, pcov_red = curve_fit(model_red, netOD_red, doses, method = "lm")
        popt_green, pcov_green = curve_fit(model_green, netOD_green, doses, method
= "lm")
        popt_blue, pcov_blue = curve_fit(model_blue, netOD_blue, doses, method =
"lm")
        popt_gray, pcov_gray = curve_fit(model_gray, netOD_gray, doses, method =
"lm")
        res_red = []; res_green = []; res_blue = []; res_gray = []

        for i in doses:
            res_red.append(netOD[np.argmin(np.abs(model_red(netOD, *popt_red) -
i))])
            res_green.append(netOD[np.argmin(np.abs(model_green(netOD, *popt_green)
- i))])
            res_blue.append(netOD[np.argmin(np.abs(model_blue(netOD, *popt_blue) -
i))])
            res_gray.append(netOD[np.argmin(np.abs(model_gray(netOD, *popt_gray) -
i))])


        rss_red = rss(doses, model_red(netOD_red, *popt_red))
        rss_green = rss(doses, model_green(netOD_green, *popt_green))
        rss_blue = rss(doses, model_blue(netOD_blue, *popt_blue))
        rss_gray = rss(doses, model_gray(netOD_gray, *popt_gray))

        rsq_red = r_squared_adjusted(doses, model_red(netOD_red, *popt_red), n=28,
p=len(popt_red))
        rsq_green = r_squared_adjusted(doses, model_green(netOD_green,
*popt_green), n=28, p=len(popt_red))
        rsq_blue = r_squared_adjusted(doses, model_blue(netOD_blue, *popt_blue),
n=28, p=len(popt_red))
        rsq_gray = r_squared_adjusted(doses, model_gray(netOD_gray, *popt_gray),
n=28, p=len(popt_red) )

        aicc_red = aicc(doses, model_red(netOD_red, *popt_red), k=len(popt_red),
n=28)
        aicc_green = aicc(doses, model_green(netOD_green, *popt_green),
k=len(popt_green), n=28)
```

```
        aicc_blue = aicc(doses, model_blue(netOD_blue,
*popt_blue),k=len(popt_blue), n=28)
        aicc_gray = aicc(doses, model_gray(netOD_gray, *popt_gray),
k=len(popt_gray), n=28)

        std_param_red = np.sqrt(np.diag(pcov_red))
        std_param_green = np.sqrt(np.diag(pcov_green))
        std_param_blue = np.sqrt(np.diag(pcov_blue))
        std_param_gray = np.sqrt(np.diag(pcov_gray))

        sigma_tot_red, sigma_fit_red, sigma_exp_red = sigmaD_red(popt_red,
netOD_red, std_param_red, std_netOD_red)
        sigma_tot_green, sigma_fit_green, sigma_exp_green =
sigmaD_green(popt_green, netOD_green, std_param_green, std_netOD_green)
        sigma_tot_blue, sigma_fit_blue, sigma_exp_blue = sigmaD_blue(popt_blue,
netOD_blue, std_param_blue, std_netOD_blue)
        sigma_tot_gray, sigma_fit_gray, sigma_exp_gray = sigmaD_gray(popt_gray,
netOD_gray, std_param_gray, std_netOD_gray)

        print "r^2  : red %.4f, green %.4f, blue %.4f, gray %.4f" %(rsq_red,
rsq_green, rsq_blue,  rsq_gray)

        def plot_models():
            r_plot, = plt.plot(model_red(netOD, *popt_red), netOD,"--r")
            r_data, = plt.plot(doses, netOD_red, ".r")
            g_plot, = plt.plot(model_green(netOD, *popt_green),netOD, "--g")
            g_data, = plt.plot(doses, netOD_green, ".g")
            b_plot, = plt.plot(model_blue(netOD, *popt_blue), netOD,"--", color =
"dodgerblue")
            b_data, = plt.plot(doses, netOD_blue, ".", color = "dodgerblue")
            m_plot, = plt.plot(model_gray(netOD, *popt_gray), netOD,"--k")
            m_data, = plt.plot(doses, netOD_gray, ".k")
            plt.xlabel("Dose [Gy]")
            plt.ylabel("netOD")
            plt.legend([r_plot,g_plot,b_plot,m_plot,
(r_data,g_data,b_data,m_data)],["fit red channel","fit green channel","fit blue
channel","fit gray channel","netOD"], handler_map={tuple:
HandlerTuple(ndivide=None)})
            plt.xlim(-1,15)
            plt.show()

    else:
        popt_red, pcov_red = curve_fit(model, netOD_red, doses, method = "lm")
        popt_green, pcov_green = curve_fit(model, netOD_green, doses, method =
"lm")
        popt_blue, pcov_blue = curve_fit(model, netOD_blue, doses, method = "lm")
        popt_gray, pcov_gray = curve_fit(model, netOD_gray, doses, method = "lm")

        for i in doses:
            res_red.append(netOD[np.argmin(np.abs(model(netOD, *popt_red) - i))])
            res_green.append(netOD[np.argmin(np.abs(model(netOD, *popt_green) -
i))])
            res_blue.append(netOD[np.argmin(np.abs(model(netOD, *popt_blue) - i))])
            res_gray.append(netOD[np.argmin(np.abs(model(netOD, *popt_gray) - i))])

        rss_red = rss(doses, model(netOD_red, *popt_red))
        rss_green = rss(doses, model(netOD_green, *popt_green))
        rss_blue = rss(doses, model(netOD_blue, *popt_blue))
        rss_gray = rss(doses, model(netOD_gray, *popt_gray))

        aicc_red = aicc(doses, model(netOD_red, *popt_red), k=len(popt_red), n=28)
        aicc_green = aicc(doses, model(netOD_green, *popt_green),
k=len(popt_green), n=28)
        aicc_blue = aicc(doses, model(netOD_blue, *popt_blue),k=len(popt_blue),
n=28)
        aicc_gray = aicc(doses, model(netOD_gray, *popt_gray), k=len(popt_gray),
n=28)

        std_param_red = np.sqrt(np.diag(pcov_red))
        std_param_green = np.sqrt(np.diag(pcov_green))
        std_param_blue = np.sqrt(np.diag(pcov_blue))
        std_param_gray = np.sqrt(np.diag(pcov_gray))

        sigma_tot_red, sigma_fit_red, sigma_exp_red  = uncertainty(popt_red,
netOD_red, std_param_red, std_netOD_red)
        sigma_tot_green, sigma_fit_green, sigma_exp_green  =
uncertainty(popt_green, netOD_green, std_param_green, std_netOD_green)
        sigma_tot_blue, sigma_fit_blue, sigma_exp_blue  = uncertainty(popt_blue,
netOD_blue, std_param_blue, std_netOD_blue)
        sigma_tot_gray, sigma_fit_gray, sigma_exp_gray  = uncertainty(popt_gray,
netOD_gray, std_param_gray, std_netOD_gray)

        def plot_models():
            r_plot, = plt.plot(model(netOD, *popt_red), netOD,"--r")
            r_data, = plt.plot(doses, netOD_red, ".r")
            g_plot, = plt.plot(model(netOD, *popt_green),netOD, "--g")
            g_data, = plt.plot(doses, netOD_green, ".g")
            b_plot, = plt.plot(model(netOD, *popt_blue), netOD,"--", color =
"dodgerblue")
            b_data, = plt.plot(doses, netOD_blue, ".", color = "dodgerblue")
            m_plot, = plt.plot(model(netOD, *popt_gray), netOD,"--k")
            m_data, = plt.plot(doses, netOD_gray, ".k")
            plt.xlabel("Dose [Gy]")
            plt.ylabel("netOD")
            plt.legend([r_plot,g_plot,b_plot,m_plot,
(r_data,g_data,b_data,m_data)],["fit red channel","fit green channel","fit blue
channel","fit gray channel","netOD"], handler_map={tuple:
HandlerTuple(ndivide=None)})
            plt.xlim(-1,15)
            plt.show()
```

```
    residual_red = netOD_red - np.array(res_red)
    residual_green = netOD_green - np.array(res_green)
    residual_blue = netOD_blue - np.array(res_blue)
    residual_gray = netOD_gray - np.array(res_gray)

    loess_red  = lowess(residual_red, doses, frac = 1)
    loess_green  = lowess(residual_green, doses, frac = 1)
    loess_blue  = lowess(residual_blue, doses, frac = 1)
    loess_gray  = lowess(residual_gray, doses, frac = 1)

    print "rss  : red %.4f, green %.4f, blue %.4f, gray %.4f" %(rss_red, rss_green,
rss_blue,  rss_gray)
    print "AICC : red %.4f, green %.4f,blue %.4f, gray %.4f" %(aicc_red,
aicc_green, aicc_blue, aicc_gray)

    print "parameters red", popt_red,std_param_red
    print "parameters green", popt_green, std_param_green
    print "parameters blue", popt_blue, std_param_blue
    print "parameters gray", popt_gray, std_param_gray

    plot_models()

    plt.rcParams["legend.handlelength"] = 5.0
    def plot_residuals():
        fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex = True, sharey =
True, gridspec_kw={'hspace': 0.1, 'wspace': 0.1})
        r_res, = ax1.plot(doses, residual_red, ".r")
        r_trend, = ax1.plot(doses[::2], loess_red[0::2,1], "--r")
        ax1.hlines(0,0,10, color = "gray")
        g_res, = ax2.plot(doses, residual_green, ".g")#
        g_trend, = ax2.plot(doses[::2], loess_green[0::2,1], "--g")
        ax2.hlines(0,0,10, color = "gray")
        b_res, = ax3.plot(doses, residual_blue, ".", color = "dodgerblue")
        b_trend, = ax3.plot(doses[::2], loess_blue[0::2,1], "--", color =
"dodgerblue")
        ax3.hlines(0,0,10, color = "gray")
        m_res, = ax4.plot(doses, residual_gray, ".k")
        m_trend, = ax4.plot(doses[::2], loess_gray[0::2,1], "--k")
        ax4.hlines(0,0,10, color = "gray")
        fig.legend([(r_res,g_res,b_res,m_res), (r_trend,g_trend,b_trend,m_trend)],
["Residuals", "Trend curves"], handler_map={tuple: HandlerTuple(ndivide=None)})
        fig.text(0.5,0.03, "Dose [Gy]", ha="center", va="center")
        fig.text(0.02,0.5, "Residuals", ha="center", va="center", rotation=90)
        plt.show()
    plot_residuals()


    def plot_uncertainties():
        fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex = True, sharey =
True, gridspec_kw={'hspace': 0.1, 'wspace': 0.1})
        ax1.plot(doses,sigma_fit_red, "^--", markersize = 3.3,color = "dimgray")
        ax1.plot(doses,sigma_exp_red, "s--", markersize = 3.3,color = "darkgray")
        totred, = ax1.plot(doses,sigma_tot_red, ".--r")
        ax2.plot(doses,sigma_fit_green, "^--", markersize = 3.3,color = "dimgray")
        ax2.plot(doses,sigma_exp_green, "s--", markersize = 3.3,color = "darkgray")
        totgreen, = ax2.plot(doses,sigma_tot_green, ".--g")
        ax3.plot(doses,sigma_fit_blue, "^--", markersize = 3.3,color = "dimgray")
        ax3.plot(doses,sigma_exp_blue, "s--", markersize = 3.3,color = "darkgray")
        totblue, = ax3.plot(doses,sigma_tot_blue, ".--", color = "dodgerblue")
        fit, = ax4.plot(doses,sigma_fit_red, "^--", markersize = 3.3,color =
"dimgray")
        exp, = ax4.plot(doses,sigma_exp_red, "s--", markersize = 3.3,color =
"darkgray")
        totgray,=  ax4.plot(doses,sigma_tot_red, ".--k")
        fig.legend([ exp, fit, (totred,totgreen,totblue,totgray)],["Experimental
uncertainty","Fit uncertainty","Total uncertainties"], handler_map={tuple:
HandlerTuple(ndivide=None)})
        fig.text(0.5,0.03, "Dose [Gy]", ha="center", va="center")
        fig.text(0.02,0.5, r"$\sigma_{Dose}$ [Gy]", ha="center", va="center",
rotation=90)
        plt.show()
    plot_uncertainties()
    plt.rcParams["legend.handlelength"] = 2.0

    #return model with lowest AICc score = best model
    if aicc_red < aicc_green and aicc_red < aicc_blue and aicc_red < aicc_gray:
        return popt_red, std_param_red, residual_red, loess_red
    if aicc_green < aicc_red and aicc_green < aicc_blue and aicc_green < aicc_gray:
        return popt_green, std_param_green, residual_green, loess_green
    if aicc_blue < aicc_red and aicc_blue < aicc_green and aicc_blue < aicc_gray:
        return popt_blue, std_param_blue, residual_blue, loess_blue
    if aicc_gray < aicc_red and aicc_gray < aicc_green and aicc_gray < aicc_blue:
        return popt_gray, std_param_gray, residual_gray, loess_gray
    return


netOD = np.linspace(0,0.5,100000)
doses = np.repeat([0.1,0.2,0.5,1.,2.,5.,10.], 4) #four films with each dose


#import  all calibration films:
exposed_films = []
black = []
path = "/Users/bjorgvh/Dropbox/Master/kalibreringsfilmer"
exposed_list =  glob.glob(os.path.join(path, "*.tif"))
unexposed_list = glob.glob(os.path.join(path+"/unexposed_films", "*.tif"))
background_list = glob.glob(os.path.join(path+"/background_films", "*.tif"))
exposed_list.sort()
unexposed_list.sort()
for file in exposed_list:
    exposed_films.append(cv2.imread(file, cv2.IMREAD_UNCHANGED))
exposed_films = np.array(exposed_films)
```

```
#plot raw data to visualize difference between experiments :

pixel_intensities =  np.zeros(len(exposed_films))
for i in range(len(exposed_films)):
    pixel_intensities[i] = np.mean(set_roi(exposed_films[i][:,:,1]))
pixel_int_unexp = np.zeros(len(unexposed_list))
for i in range(len(unexposed_list)):
    pixel_int_unexp[i]  = np.mean(set_roi(cv2.imread(unexposed_list[i],
cv2.IMREAD_UNCHANGED)[:,:,1]))

plt.plot(doses,pixel_intensities, ".", color = "tan", label = "Experiment 1")
plt.plot(doses[2::4],pixel_intensities[2::4], ".",color = "darkslategray",label =
"Experiment 2")
plt.plot(doses[3::4],pixel_intensities[3::4], ".",color = "darkslategray")
plt.plot([0,0], pixel_int_unexp[2:], ".", color ="darkslategray")
plt.plot([0,0], pixel_int_unexp[:2], ".", color ="tan")
plt.legend()
plt.ylabel("Pixel intensity")
plt.xlabel("Dose [Gy]")
plt.title("Measured pixel intensity vs given dose for calibratiton films")
plt.show()



#find i_unexp og i_bckg
unexp_red, unexp_green, unexp_blue, unexp_gray, sigma_unexp_red,
sigma_unexp_green,sigma_unexp_blue, sigma_unexp_gray, = i_unexposed(unexposed_list)
bckg_red, bckg_green, bckg_blue, bckg_gray, sigma_bckg_red, sigma_bckg_green,
sigma_bckg_blue, sigma_bckg_gray= i_unexposed(background_list)
print unexp_green, bckg_green


#find netOD in ROI for all calibration films
netOD_red = np.zeros(len(exposed_films)); netOD_green =
np.zeros(len(exposed_films)); netOD_blue = np.zeros(len(exposed_films)); netOD_gray
= np.zeros(len(exposed_films))
std_netOD_red = np.zeros(len(exposed_films)); std_netOD_green =
np.zeros(len(exposed_films)); std_netOD_blue = np.zeros(len(exposed_films));
std_netOD_gray = np.zeros(len(exposed_films))
for i in range (len(exposed_films)):
    gray_img = cv2.cvtColor(exposed_films[i], cv2.COLOR_BGR2GRAY)

    roi_red = set_roi(exposed_films[i][:,:,2])
    roi_green = set_roi(exposed_films[i][:,:,1])
    roi_blue = set_roi(exposed_films[i][:,:,0])
    roi_gray  = set_roi(gray_img)

    netOD_red[i], std_netOD_red[i] = find_netOD(roi_red, unexp_red, bckg_red,
sigma_unexp_red, sigma_bckg_red)
```

```
    netOD_green[i], std_netOD_green[i] = find_netOD(roi_green, unexp_green,
bckg_green, sigma_unexp_green, sigma_bckg_green)
    netOD_blue[i], std_netOD_blue[i] = find_netOD(roi_blue, unexp_blue, bckg_blue,
sigma_unexp_blue, sigma_bckg_blue)
    netOD_gray[i], std_netOD_gray[i] = find_netOD(roi_gray, unexp_gray, bckg_gray,
sigma_unexp_gray, sigma_bckg_gray)

netOD_red[netOD_red<0] = 0
netOD_green[netOD_green<0] = 0
netOD_blue[netOD_blue<0] = 0
netOD_gray[netOD_gray<0] = 0



# test models:

print "--------------------------------------"
print "model 1"
popt_model1, std_param_model1, res_model1, loess_model1 = run_model(netOD_red,
netOD_green, netOD_blue, netOD_gray, find_model1, find_model1, std_netOD_red,
std_netOD_green, std_netOD_blue, std_netOD_gray, False)
print "--------------------------------------"
print "model 2"
popt_model2, std_param_model2, res_model2, loess_model2 = run_model(netOD_red,
netOD_green, netOD_blue, netOD_gray, model2, sigma_Dmodel2, std_netOD_red,
std_netOD_green, std_netOD_blue, std_netOD_gray)
print "--------------------------------------"
print "model 3"
popt_model3, std_param_model3, res_model3, loess_model3 = run_model(netOD_red,
netOD_green, netOD_blue, netOD_gray, model3, sigma_Dmodel3, std_netOD_red,
std_netOD_green, std_netOD_blue, std_netOD_gray)
print "--------------------------------------"
print "model 4"
popt_model4, std_param_model4, res_model4, loess_model4 = run_model(netOD_red,
netOD_green, netOD_blue, netOD_gray, find_model4, find_model4, std_netOD_red,
std_netOD_green, std_netOD_blue, std_netOD_gray, False)


#Model 4, gren channel is found to be best model:
model4, Dtot_model4 = find_model4(doses, netOD_green, "green")
total_uncert, fit_uncert, exp_uncert = Dtot_model4(popt_model4, netOD_green,
std_param_model4, std_netOD_green)


#evaluation of p-values not possible through curve_fit, use statsmodel
#(is verified to give same parameters and uncertainties)
import statsmodels.api as sm
X = netOD_green**1.25
y = doses
model = sm.OLS(y, X).fit()
print model.pvalues
```

```
X = np.stack((netOD_green, netOD_green**1.5), axis = -1)
model = sm.OLS(y, X).fit()
print model.pvalues

X = netOD_red**1.5
model = sm.OLS(y, X).fit()
print model.pvalues
X = np.stack((netOD_red, netOD_red**2.), axis = -1)
model = sm.OLS(y, X).fit()
print model.pvalues

X = netOD_blue**1.25
model = sm.OLS(y, X).fit()
print model.pvalues
X = np.stack((netOD_blue, netOD_blue**0.75), axis = -1)
model = sm.OLS(y, X).fit()
print model.pvalues

X = netOD_gray**1.25
model = sm.OLS(y, X).fit()
print model.pvalues
X = np.stack((netOD_gray, netOD_gray**1.5), axis = -1)
model = sm.OLS(y, X).fit()
print model.pvalues




#interpolate to find uncertainnty function for all netOD
dosepoints = np.array([0.1,0.2,0.5,1,2,5,10])
netOD_mean = np.mean(np.reshape(netOD_green, (7,4)), axis = 1)
std_netOD_mean = np.std(np.reshape(netOD_green, (7,4)), axis = 1)/np.sqrt(4.)

total_uncertainty, fit, exp = Dtot_model4(popt_model4, netOD_mean,
std_param_model4, std_netOD_mean)

sigma_Dtot = interpolate.UnivariateSpline(dosepoints, total_uncertainty)
x = np.linspace(0.1,10,10000)
uncert_array = np.stack((x, sigma_Dtot(x)), axis = -1)
np.savetxt("/Users/bjorgvh/Dropbox/Master/doseuncertainty.txt",uncert_array)


#PLOT div.

plt.plot(x,sigma_Dtot(x), "-g", label = "Spline interpolation")
plt.plot(dosepoints, total_uncertainty, "og", label = "Mean total uncertainty per
dose")
plt.legend()
plt.ylabel(r"$\sigma_{\overline{Dose}}$ [Gy]")
plt.xlabel("Dose [Gy]")
plt.ylim(-0.02,0.55)
plt.show()
plt.plot(doses,fit_uncert, "^--",markersize = 3.3,color = "dimgray", label = "Fit
uncertainty")
plt.plot(doses,exp_uncert, "s--",markersize = 3.3,color = "darkgray", label =
"Experimental uncertainty")
plt.plot(doses,total_uncert, ".--g", label = "Total uncertanty")
plt.legend()
plt.ylabel(r"$\sigma_{Dose}$ [Gy]")
plt.xlabel("Dose [Gy]")
plt.ylim(-0.02,0.55)
plt.show()

plt.plot(doses[0::4], res_model1[0::4], ".g", label = "Residuals experiment 1")
plt.plot(doses[0::4], res_model1[1::4], ".g")
plt.plot(doses[0::4], res_model1[2::4], "^g", markersize = 3.3,  label = "Residuals
experiment 2")
plt.plot(doses[0::4], res_model1[3::4], "^g",markersize = 3.3)
plt.plot(doses[::2], loess_model1[0::2,1], "--g", label = "Trend curve")
plt.hlines(0,0,10, color = "gray")
plt.legend()
plt.xlabel("Dose [Gy]")
plt.show()

plt.plot(doses[0::4], netOD_green[0::4], ".g")
plt.plot(doses[1::4], netOD_green[1::4], ".g", label = "netOD green channel, exp
1")
plt.plot(doses[2::4], netOD_green[2::4], "^g", markersize = 3.3)
plt.plot(doses[3::4], netOD_green[3::4], "^g",markersize = 3.3, label = "netOD
green channel, exp 2")
plt.plot(model4(netOD, *popt_model4), netOD, "--g", label = r"$ a \cdot
netOD^{1.25}$")
plt.legend()
plt.xlabel("Dose [Gy]")
plt.ylabel("netOD")
plt.show()

plt.plot(0,0,".", label  = "Red channel:   ", markersize = 0)
plt.plot(0,0,".", label  = "Green channel: ", markersize = 0)
plt.plot(0,0,".", label  = "Blue channel:  ", markersize = 0)
plt.plot(0,0,".", label  = "Grayscale image:", markersize = 0)
plt.plot(doses[0::4], netOD_red[0::4], ".r", label = "experiment 1")
plt.plot(doses[0::4], netOD_red[1::4], ".r")
plt.plot(doses[0::4], netOD_green[0::4], ".g",label = "experiment 1")
plt.plot(doses[0::4], netOD_green[1::4], ".g")
plt.plot(doses[0::4], netOD_blue[0::4], ".", color = "dodgerblue",label =
"experiment 1")
plt.plot(doses[0::4], netOD_blue[1::4], ".", color = "dodgerblue")
plt.plot(doses[0::4], netOD_gray[0::4], ".k",label = "experiment 1")
plt.plot(doses[0::4], netOD_gray[1::4], ".k")
plt.plot(doses[2::4], netOD_red[2::4], "^r",markersize = 3.3, label = "experiment
2")
```

```
plt.plot(doses[2::4], netOD_red[3::4], "^r", markersize = 3.3)
plt.plot(doses[2::4], netOD_green[2::4], "^g",markersize = 3.3,label = "experiment
2")
plt.plot(doses[2::4], netOD_green[3::4], "^g",markersize = 3.3)
plt.plot(doses[2::4], netOD_blue[2::4], "^",markersize = 3.3, color =
"dodgerblue",label = "experiment 2")
plt.plot(doses[2::4], netOD_blue[3::4], "^", markersize = 3.3,color = "dodgerblue")
plt.plot(doses[2::4], netOD_gray[2::4], "^k",markersize = 3.3,label = "experiment
2")
plt.plot(doses[2::4], netOD_gray[3::4], "^k",markersize = 3.3)
lgnd = plt.legend(ncol=3, columnspacing = 0.01, markerscale= 0,handletextpad=0.01)
for i in range(4,8):
    lgnd.legendHandles[i]._legmarker.set_markersize(7)
for i in range(8,12):
    lgnd.legendHandles[i]._legmarker.set_markersize(3.3)
plt.xlabel("Dose [Gy]")
plt.ylabel("netOD")
plt.show()
```

GRID films X-ray

```
# find dose profiles from GRID

import numpy as np
import matplotlib.pyplot as plt
import cv2
import glob
import os
import scipy.signal
import csv

def dose(i_exp): #best fit from calibration
    i_unexp = 29196.331339369095 #green channel
    i_bckg =  608.4299547416639 #green channel
    netOD = np.log10((i_unexp-i_bckg)/(i_exp-i_bckg))
    return np.nan_to_num(28.35820601*netOD**1.25)

def profile(image):
    profile = np.zeros(image.shape[0])
    for i in range(image.shape[0]):
        profile[i] = np.sum(image[i])/np.sum(image[i]>0)
    return np.nan_to_num(profile)

def profile_per_dx(prof, wanted_dx):
    length = 3000 # all images is 3000 pixels long
    pixel_to_cm = 1./(1200./2.54)  # 1 inch = 2.54 cm, 1200 dpi image
    cm_to_pixel = 1200./2.54
    dx_pixel = int(np.round(wanted_dx * cm_to_pixel)) #get dx in pixels

    n_bands = int(np.ceil(float(length)/dx_pixel))
    new_length = dx_pixel * n_bands
    new_prof = np.append(prof, np.zeros(new_length - length))

    reshaped = np.reshape(new_prof, (n_bands,-1)) #shape into bands of dx
    prof = np.mean(reshaped, axis =1)
    return prof, np.linspace(0,length*pixel_to_cm,n_bands)


length = 3000 # all images is 3000 pixels long
pixel_to_cm = 1./(1200./2.54)  # 1 inch = 2.54 cm, 1200 dpi image
cm = np.linspace(0,length*pixel_to_cm,length)
path  = "/Users/bjorgvh/Dropbox/Master/Gridfilm"
filelist = glob.glob(os.path.join(path, "*.tif"))
filelist.sort()
correction_vector = np.array(open("/Users/bjorgvh/Dropbox/Master/diff_vector.txt",
"r").readlines(), dtype = float)
```

```
#Example imange of dosemap
img = cv2.imread("/Users/bjorgvh/Dropbox/Master/Gridfilm/posC2-dag2-crop.tif",
cv2.IMREAD_UNCHANGED)[:,:,1]
doses = dose(img)
plt.imshow(doses, vmin = 0, vmax = 5)
plt.colorbar()
plt.show()


#plot raw  data
pixel_intensities = []
for i in range(len(filelist)):
    imgs = cv2.imread(filelist[i], cv2.IMREAD_UNCHANGED)[:,:,2] #red channel
    pixel_intensities.append(profile(imgs))
pixel_intensities = np.reshape(pixel_intensities,(16,imgs.shape[0]))

for i in range(pixel_intensities.shape[0]):
    if filelist[i][-13:-9] == "dag1":
        plt.plot(cm, pixel_intensities[i], color = "tan")
    else:
        plt.plot(cm, pixel_intensities[i], color = "darkslategray")
plt.legend(["Experiment 1", "Experiment 2"])
plt.ylabel("Pixel intensity")
plt.xlabel("Position in flask [cm]")
plt.title("Pixel intensity profiles from X-ray GRID films")
plt.show()


#find dose profiles for all GRID-films
profiles = []
for i in range(len(filelist)):
    img = cv2.imread(filelist[i], cv2.IMREAD_UNCHANGED)[:,:,1] #green channel
    dosemap = dose(img)
    profiles.append(profile(dosemap))
profiles = np.reshape(profiles,(16,img.shape[0]))

profiles = np.delete(profiles, 12, axis =0) # delete the abnormal dose
del filelist[12]

for i in range(profiles.shape[0]):
    if filelist[i][-13:-9] == "dag1":
        plt.plot(cm, profiles[i],"tan")
    else:
        plt.plot(cm, profiles[i], "darkslategrey")
plt.plot(0,0, color ="tan",label = "Experiment 1, GRID")
plt.plot(0,0, color ="darkslategrey",label = "Experiment 2, GRID")


# find profile of open field-films
openlist = glob.glob(os.path.join("/Users/bjorgvh/Dropbox/Master/Gridfilm/open",
"*.tif"))
openlist.sort()
```

```
open_field = []
for file in openlist:
    img = cv2.imread(file, cv2.IMREAD_UNCHANGED)[:,:,1]  #green channel
    doses= dose(img)
    open_field.append(profile(doses))
open_field = np.reshape(open_field,(6,img.shape[0]))

plt.plot(cm[300:2800], open_field[0][300:2800],color ="tan", dashes=(5,5), label =
"Experiment 1, open field")
plt.plot(cm[300:2800], open_field[1][300:2800],color ="tan",dashes=(5,5))
plt.plot(cm[300:2800], open_field[2][300:2800],color
="darkslategrey",dashes=(5,5),label = "Experiment 2, open field")
plt.plot(cm[300:2800], open_field[3][300:2800],color ="darkslategrey",dashes=(5,5))

plt.legend(loc = (0,0.25))
plt.title("Dose profiles from X-ray GRID films")
plt.ylabel("Dose [Gy]")
plt.xlabel("Position in flask [cm]")
plt.show()


#shift all GRID-films to match median using phase correlation, and do correction
for gradient
median = np.median(profiles, axis = 0)
shifted_profiles = np.zeros(profiles.shape)
f1 = np.fft.fft(median)
for i in range(profiles.shape[0]):

    f0 = np.fft.fft(profiles[i])
    correlation = np.abs(np.fft.ifft((f0 * np.conj(f1))))
    shift = np.unravel_index(np.argmax(correlation), f0.shape)[0]

    if shift > f1.shape[0] // 2:
        shift -= f1.shape[0]

    if shift > 0:
        shifted_img = np.zeros(len(profiles[i]) + shift)
        shifted_img[:-shift] = profiles[i]
        shifted_profiles[i] = shifted_img[shift:]
    if shift < 0:
        shift = np.abs(shift)
        shifted_img = np.zeros(len(profiles[i]) + shift)
        shifted_img[shift:] = profiles[i]
        shifted_profiles[i] = shifted_img[:-shift]
    if shift == 0:
        shifted_profiles[i] = profiles[i]
    shifted_profiles[i] -= correction_vector

for i in range(shifted_profiles.shape[0]):
    if filelist[i][-13:-9] == "dag1":
        plt.plot(cm, shifted_profiles[i],"tan")
    else:
        plt.plot(cm, shifted_profiles[i], "darkslategrey")
plt.plot(0,0, color ="tan",label = "Experiment 1, GRID")
plt.plot(0,0, color ="darkslategrey",label = "Experiment 2, GRID")

open_field -= correction_vector
plt.plot(cm[300:2800], open_field[0][300:2800],color ="tan", dashes=(5,5), label =
"Experiment 1, open field")
plt.plot(cm[300:2800], open_field[1][300:2800],color ="tan",dashes=(5,5))
plt.plot(cm[300:2800], open_field[2][300:2800],color
="darkslategrey",dashes=(5,5),label = "Experiment 2, open field")
plt.plot(cm[300:2800], open_field[3][300:2800],color ="darkslategrey",dashes=(5,5))

plt.legend(loc = (0,0.25))
plt.title("Dose profiles from X-ray GRID films, corrected for gradient and shifted
horizontally to match.")
plt.ylabel("Dose [Gy]")
plt.xlabel("Position in flask [cm]")
plt.show()


#find mean dose profile, with 95% confidence band
mean_profile = np.mean(shifted_profiles, axis = 0)
mean_profile10gy = 2*mean_profile
mean_profile2gy = 2./5*mean_profile
mean_open_field = np.mean(open_field[:4], axis = 0)
meanGRID2gy = np.mean(mean_profile2gy[300:2800])
meanGRID5gy = np.mean(mean_profile[300:2800])
meanGRID10gy = np.mean(mean_profile10gy[300:2800])


#list of uncertainty in doses from fit from 0.1 to 10 Gy, 10000 steps:
dosepoint= []; doseuncert =[]
with open("/Users/bjorgvh/Dropbox/Master/doseuncertainty.txt", "r") as file:
    for line in file:
        data = line.split()
        dosepoint.append(data[0]); doseuncert.append(data[1])
sigma_Dfit_list = np.stack((np.array(dosepoint, dtype = float),np.array(doseuncert,
dtype = float)), axis = -1)

#find uncertainty of mean profile and open field profile
sigma_Dfit_meanprofile = np.zeros(len(mean_profile))
sigma_Dfit_meanprofile10gy = np.zeros(len(mean_profile))
sigma_Dfit_meanprofile2gy = np.zeros(len(mean_profile))
sigma_Dfit_openfield = np.zeros(len(mean_profile))

for i in range(len(mean_profile)):
    sigma_Dfit_meanprofile[i] =
sigma_Dfit_list[np.argmin(np.abs(sigma_Dfit_list[:,0]-mean_profile[i])), 1]
    sigma_Dfit_meanprofile10gy[i] =
sigma_Dfit_list[np.argmin(np.abs(sigma_Dfit_list[:,0]-mean_profile10gy[i])), 1]
```

```
    sigma_Dfit_meanprofile2gy[i] =
sigma_Dfit_list[np.argmin(np.abs(sigma_Dfit_list[:,0]-mean_profile2gy[i])), 1]
    sigma_Dfit_openfield[i] =
sigma_Dfit_list[np.argmin(np.abs(sigma_Dfit_list[:,0]-mean_open_field[i])), 1]

sigma_GRID = np.std(shifted_profiles, axis = 0)
conf_band = 1.96*np.sqrt(sigma_Dfit_meanprofile**2 +
sigma_GRID**2/np.sqrt(profiles.shape[0]))
conf_band10gy = 1.96*np.sqrt(sigma_Dfit_meanprofile10gy**2 +
sigma_GRID**2/np.sqrt(profiles.shape[0]))
conf_band2gy = 1.96*np.sqrt(sigma_Dfit_meanprofile2gy**2 +
sigma_GRID**2/np.sqrt(profiles.shape[0]))
conf_band_open_field = (1.96*np.sqrt(sigma_Dfit_openfield**2 +
np.std(open_field[:4], axis = 0)**2/np.sqrt(4)))

print "open field ", np.mean(mean_open_field[300:2800]), "+/-" ,
np.mean(conf_band_open_field[300:2800])
print "Peak dose ", np.mean((mean_profile[825:950], mean_profile[1575:1700],
mean_profile[2325:2450])),"+/-" , np.mean((conf_band[825:950],
conf_band[1575:1700], conf_band[2325:2450]))
print "Valley dose ", np.mean((mean_profile[300:700], mean_profile[1050:1450],
mean_profile[1800:2200])),"+/-" , np.mean((conf_band[300:700],
conf_band[1050:1450], conf_band[1800:2200]))
print "mean GRID 2gy ", meanGRID2gy
print "mean GRID 5gy ", meanGRID5gy
print "mean GRID 10gy ", meanGRID10gy

plt.fill_between(cm[300:2800], mean_profile[300:2800]-conf_band[300:2800],
mean_profile[300:2800]+conf_band[300:2800], color = "0.75", label = "95% confidence
interval")
plt.plot(cm[300:2800], mean_profile[300:2800], "-k", label = "Mean profile, GRID")
plt.fill_between(cm[300:2800], mean_open_field[300:2800]-
conf_band_open_field[300:2800],
mean_open_field[300:2800]+conf_band_open_field[300:2800], color = "0.75")
plt.plot(cm[300:2800], mean_open_field[300:2800], "k", dashes=(5,5), label = "Mean
profile, open field")
plt.ylabel("Dose [Gy]")
plt.xlabel("Position in flask [cm]")
plt.title("Mean dose profile for X-ray GRID films")
plt.legend(loc = (0,0.25))
plt.show()


# Saving dose profiles per band width, same dx as when counting cells
mean_profile_dx05, x05_dose = profile_per_dx(mean_profile, 0.05)
mean_profile_dx15, x15_dose = profile_per_dx(mean_profile, 0.15)
conf_band_dx05,x = profile_per_dx(conf_band, 0.05)
conf_band_dx15,x = profile_per_dx(conf_band, 0.15)

mean_profile_dx05_10gy, x05_dose = profile_per_dx(mean_profile10gy, 0.05)
mean_profile_dx15_10gy, x15_dose = profile_per_dx(mean_profile10gy, 0.15)
```

```
conf_band_dx05_10gy,x = profile_per_dx(conf_band10gy, 0.05)
conf_band_dx15_10gy,x = profile_per_dx(conf_band10gy, 0.15)

mean_profile_dx05_2gy, x05_dose = profile_per_dx(mean_profile2gy, 0.05)
mean_profile_dx15_2gy, x15_dose = profile_per_dx(mean_profile2gy, 0.15)
conf_band_dx05_2gy,x = profile_per_dx(conf_band2gy, 0.05)
conf_band_dx15_2gy,x = profile_per_dx(conf_band2gy, 0.15)

np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose.txt", mean_profile)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx05.txt",
mean_profile_dx05)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx15.txt",
mean_profile_dx15)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/conf_band.txt", conf_band)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx05.txt",
conf_band_dx05)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx15.txt",
conf_band_dx15)

np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose10gy.txt",
mean_profile10gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx05_10gy.txt",
mean_profile_dx05_10gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx15_10gy.txt",
mean_profile_dx15_10gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/conf_band10gy.txt",
conf_band10gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx05_10gy.txt",
conf_band_dx05_10gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx15_10gy.txt",
conf_band_dx15_10gy)

np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose2gy.txt",
mean_profile2gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx05_2gy.txt",
mean_profile_dx05_2gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx15_2gy.txt",
mean_profile_dx15_2gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/conf_band2gy.txt",
conf_band2gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx05_2gy.txt",
conf_band_dx05_2gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx15_2gy.txt",
conf_band_dx15_2gy)


# Test of film with nylon6 covering half
img = cv2.imread("/Users/bjorgvh/Dropbox/Master/scan_film_rontgen_17des/dag2-
halv.tif", cv2.IMREAD_UNCHANGED)[500:2800,500:1700,1]
doses = dose(img)
halv = profile(doses) - correction_vector[500:2800]
```

```
print np.mean(halv[:1000]), np.mean(halv[1200:])
print (np.mean(halv[1200:])-np.mean(halv[:1000]))/np.mean(halv[1200:])
plt.plot(cm[500:2800],halv)
plt.ylabel("Dose [Gy]")
plt.xlabel("Position in flask [cm]")
plt.title("Profile of EBT3 film with Nylon6 over last half")
plt.show()


#saving all dose maps of GRID films as txt + image.
path  = "/Users/bjorgvh/Dropbox/Master/Gridfilm"
filelist = glob.glob(os.path.join(path, "*.tif"))
for filename in sorted(filelist):
    img = cv2.imread(filename, cv2.IMREAD_UNCHANGED)[:,:,2]
    doses = dose(img)
    name = filename[-19:-9]
    np.savetxt("/Users/bjorgvh/Dropbox/Master/%s.txt" %name, doses)
    plt.imsave("/Users/bjorgvh/Dropbox/Master/%s.png" %name, doses)
```

Find correction for gradient

```
#check gradient from scanner with calibration films

import numpy as np
import matplotlib.pyplot as plt
import cv2
import glob
import os
from matplotlib.legend_handler import HandlerTuple
from scipy import stats
from scipy.interpolate import UnivariateSpline

def dose(roi):
    i_exp = roi
    i_unexp = 29196.331339369095 #red channel
    i_bckg =  608.4299547416639  #red channel
    netOD = np.log10((i_unexp-i_bckg)/(i_exp-i_bckg))
    return np.nan_to_num(28.35820601*netOD**1.25)

profiles = np.zeros((34,1300))
files = glob.glob(os.path.join("/Users/bjorgvh/Dropbox/Master/kalibreringsfilmer/",
"*.tif"))
files.sort()
for i in range(28):
    img = cv2.imread(files[i], cv2.IMREAD_UNCHANGED)[400:1700,400:1600,1]
    dosemap = dose(img)
    profiles[i] = np.mean(dosemap, axis = 1)

unexp_img1 =
cv2.imread("/Users/bjorgvh/Dropbox/Master/kalibreringsfilmer/unexposed_films/dose1-
0gy-dag2-2.tif", cv2.IMREAD_UNCHANGED)[400:1700,400:1600,1]
unexp_img2 =
cv2.imread("/Users/bjorgvh/Dropbox/Master/kalibreringsfilmer/unexposed_films/dose1-
0gy-dag2-3.tif", cv2.IMREAD_UNCHANGED)[400:1700,400:1600,1]
unexp1 = np.mean(dose(unexp_img1), axis = 1)
unexp2 = np.mean(dose(unexp_img2), axis = 1)

cm = np.linspace(0,1300./(1200./2.54),1300)

fig, ((ax1,ax2),(ax3,ax4),(ax5,ax6),(ax7,ax8)) = plt.subplots(4,2, sharex = True)
ax1.plot(cm, unexp1, label = "0 Gy",color="darkkhaki")
ax1.plot(cm, unexp2,color = "darkolivegreen")
ax2.plot(cm, profiles[2],label = "0.1 Gy",color="darkkhaki")
ax2.plot(cm, profiles[3],color="darkolivegreen")
ax3.plot(cm, profiles[6],label = "0.2 Gy",color="darkkhaki")
ax3.plot(cm, profiles[7],color="darkolivegreen")
ax4.plot(cm, profiles[10],label = "0.5 Gy",color="darkkhaki")
ax4.plot(cm, profiles[11],color="darkolivegreen")
```

```
ax5.plot(cm, profiles[14],label = "1 Gy",color="darkkhaki")
ax5.plot(cm, profiles[15],color="darkolivegreen")
ax6.plot(cm, profiles[18],label = "2 Gy",color="darkkhaki")
ax6.plot(cm, profiles[19],color="darkolivegreen")
ax7.plot(cm, profiles[22],label = "5 Gy",color="darkkhaki")
ax7.plot(cm, profiles[23],color="darkolivegreen")
ax8.plot(cm, profiles[26],label = "10 Gy",color="darkkhaki")
ax8.plot(cm, profiles[27],color="darkolivegreen")

ax1.legend(handlelength=0, loc = "lower right")
ax2.legend(handlelength=0, loc = "lower right")
ax3.legend(handlelength=0, loc = "lower right")
ax4.legend(handlelength=0, loc = "lower right")
ax5.legend(handlelength=0, loc = "lower right")
ax6.legend(handlelength=0, loc = "lower right")
ax7.legend(handlelength=0, loc = "lower right")
ax8.legend(handlelength=0, loc = "lower right")

plt.suptitle("Dose profiles from calibration films, experiment 2")
fig.text(0.02, 0.53, "Dose [Gy]", rotation="vertical")
fig.text(0.4, 0.03, "Position in flask [cm]")
plt.show()

dose5gydag1_1 = np.mean(dose(cv2.imread(files[20], cv2.IMREAD_UNCHANGED)[400:1700,
400:1900,1]),axis = 1)
dose5gydag1_2 = np.mean(dose(cv2.imread(files[21], cv2.IMREAD_UNCHANGED)[400:1700,
400:1900,1]),axis = 1)
dose5gydag2_1 = np.mean(dose(cv2.imread(files[22], cv2.IMREAD_UNCHANGED)[400:1700,
400:1900,1]),axis = 1)
dose5gydag2_2 = np.mean(dose(cv2.imread(files[23], cv2.IMREAD_UNCHANGED)[400:1700,
400:1900,1]),axis = 1)

#find gradient and save corraction vector
x1 = np.arange(1700)[400:]
slope1, intersect1, r_value1, p_value1, std_err1= stats.linregress(x1,
dose5gydag1_1)
slope2, intersect2,r_value2, p_value2, std_err2 = stats.linregress(x1,
dose5gydag1_2)
slope3, intersect3, r_value1, p_value1, std_err1= stats.linregress(x1,
dose5gydag2_1)
slope4, intersect4, r_value2, p_value2, std_err2 = stats.linregress(x1,
dose5gydag2_2)

baseline = np.ones(3000)*5.
x = np.arange(3000)

diff1 = (slope1*x+intersect1) - baseline
diff2 = (slope2*x+intersect2) - baseline
diff3 = (slope3*x+intersect3) - baseline
diff4 = (slope4*x+intersect4) - baseline
diff = np.mean((diff1,diff2,diff3,diff4), axis = 0)
```

```
np.savetxt("diff_vector.txt", diff)
```

Redo calibration with old X-ray scan

```
#use old X-ray scan to get calibration for proton films, compare with new X-ray
scan

import numpy as np
import matplotlib.pyplot as plt
import cv2
import glob
import os
from scipy.optimize import curve_fit
from scipy import interpolate

def dose(roi):
    i_exp = np.mean(roi)
    i_unexp = 29196.331339369095#red channel
    i_bckg =  608.4299547416639 #red channel
    netOD = np.log10((i_unexp-i_bckg)/(i_exp-i_bckg))
    return np.nan_to_num(28.35820601*netOD**1.25), netOD

def netOD_old_scan(roi):
    i_exp = np.mean(roi)
    sigma_exp = np.std(roi)
    netOD = np.log10((i_unexp-i_bckg)/(i_exp-i_bckg))
    sigma_netOD = np.sqrt((sigma_unexp + sigma_bckg)/(i_unexp - i_bckg)**2 +
(sigma_exp**2 + sigma_bckg)/(i_exp - i_bckg)**2)
    return netOD, sigma_netOD

def model(netOD,a):
    return a*netOD**1.25

def uncert_Dtot(a, netOD, sigma_a, sigma_netOD):
    n=1.25
    return np.sqrt(netOD**(2*n) * sigma_a**2 + (a*n*netOD**(n-1))**2 *
sigma_netOD**2)

def uncert_Dfit(netOD, sigma_a): #fit uncertainty of dose fit
    n=1.25
    return np.sqrt(netOD**(2*n) * sigma_a**2)

def uncert_Dexp(a, netOD, sigma_netOD): #experimental uncertainty of dose fit
    n=1.25
    return np.sqrt((a*n*netOD**(n-1))**2 * sigma_netOD**2)

def i_unexp_bckg(list):
    num = 0
    denom = 0
    for file in list:
        unexp = cv2.imread(file, cv2.IMREAD_UNCHANGED)[:,:,1]
        min = unexp.shape[1]/2 - 24
        max = unexp.shape[1]/2 + 24
        roi = unexp[min:max,min:max]
        num += np.mean(roi)/np.std(roi)**2
        denom += 1./np.std(roi)**2
    return num/denom, 1/denom

# scan used for comparing films to cells, true scan:
scan2 = glob.glob(os.path.join("/Users/bjorgvh/Dropbox/Master/kalibreringsfilmer",
"*.tif"))
scan2.sort()

scan2_exp1 = scan2[0::4] + scan2[1::4]; scan2_exp1.sort()
scan2_exp2 = scan2[2::4] + scan2[3::4]; scan2_exp2.sort()

scan2_exp1_dose= np.zeros(len(scan2_exp1))
scan2_exp1_netOD = np.zeros(len(scan2_exp1))
scan2_exp2_dose = np.zeros(len(scan2_exp2))
scan2_exp2_netOD = np.zeros(len(scan2_exp2))

for i in range(len(scan2_exp1)):
    film_exp1 = cv2.imread(scan2_exp1[i], cv2.IMREAD_UNCHANGED)[:,:,1]
    film_exp2 = cv2.imread(scan2_exp2[i], cv2.IMREAD_UNCHANGED)[:,:,1]

    min = film_exp1.shape[0]/2 - 95
    max = film_exp1.shape[0]/2 + 95

    scan2_exp1_dose[i], scan2_exp1_netOD [i] = dose(film_exp1[min:max,min:max])
    scan2_exp2_dose[i], scan2_exp2_netOD [i] = dose(film_exp2[min:max,min:max])

scan2_netOD =
np.append(np.reshape(scan2_exp1_netOD,(7,2)),np.reshape(scan2_exp2_netOD,(7,2)),
axis = 1)
scan2_dose =
np.append(np.reshape(scan2_exp1_dose,(7,2)),np.reshape(scan2_exp2_dose,(7,2)), axis
= 1)

#old scan of X-ray films:
scan1 =
glob.glob(os.path.join("/Users/bjorgvh/Dropbox/Master/scan_film_rontgen_okt-nov",
"*.tif"))
scan1.sort()

scan1_exp1 = []
scan1_exp2 = []

for file in scan1:
    if file[-20:-16]  == "dag1" or file[-19:-15] == "dag1":
        scan1_exp1.append(file)
    if file[-20:-16]  == "dag2" or file[-19:-15] == "dag2":
        scan1_exp2.append(file)
scan1_exp1.sort()
```

```
scan1_exp2.sort()

unexposed = scan1_exp1[:2] + scan1_exp2[:2]
background = scan1[3:5]

i_unexp, sigma_unexp = i_unexp_bckg(unexposed)
i_bckg, sigma_bckg = i_unexp_bckg(background)
print i_unexp, i_bckg

scan1_exp1_dose = np.zeros(len(scan1_exp1)-2)
scan1_exp1_netOD = np.zeros(len(scan1_exp1)-2)
scan1_exp1_sigma_netOD = np.zeros(len(scan1_exp1)-2)
scan1_exp2_dose = np.zeros(len(scan1_exp2)-2)
scan1_exp2_netOD = np.zeros(len(scan1_exp2)-2)
scan1_exp2_sigma_netOD = np.zeros(len(scan1_exp2)-2)

for i in range(2,len(scan1_exp1)):
    img1 = cv2.imread(scan1_exp1[i], cv2.IMREAD_UNCHANGED)[:,:,1]
    img2 = cv2.imread(scan1_exp2[i], cv2.IMREAD_UNCHANGED)[:,:,1]

    min1 = img1.shape[0]/2 - 24
    max1 = img1.shape[0]/2 + 24
    min2 = img2.shape[0]/2 - 24
    max2 = img2.shape[0]/2 + 24

    scan1_exp1_netOD[i-2], scan1_exp1_sigma_netOD[i-2]=
netOD_old_scan(img1[min1:max1,min1:max1])
    scan1_exp2_netOD[i-2], scan1_exp2_sigma_netOD[i-2]=
netOD_old_scan(img2[min2:max2,min2:max2])
scan1_netOD = np.ravel(np.append(np.reshape(scan1_exp1_netOD,
(7,2)),np.reshape(scan1_exp2_netOD, (7,2)), axis = 1))
scan1_sigma_netOD = np.ravel(np.append(np.reshape(scan1_exp1_sigma_netOD,
(7,2)),np.reshape(scan1_exp2_sigma_netOD, (7,2)), axis = 1))


dosepoints_fit = np.repeat([0.1,0.2,0.5,1,2,5,10],4)
popt, pcov = curve_fit(model, scan1_netOD, dosepoints_fit, method = "lm")
sigma_param = np.sqrt(np.diag(pcov))
print popt, sigma_param

sigma_tot = uncert_Dtot(popt, scan1_netOD, sigma_param, scan1_sigma_netOD)
sigma_exp = uncert_Dexp(popt, scan1_netOD, scan1_sigma_netOD)
sigma_fit = uncert_Dfit(scan1_netOD, sigma_param)

plt.plot(dosepoints_fit, sigma_fit, "^--",markersize = 3.3,color = "dimgray", label
= "Fit uncertainty")
plt.plot(dosepoints_fit, sigma_exp, "s--",markersize = 3.3,color = "darkgray",
label = "Experimental uncertainty")
plt.plot(dosepoints_fit, sigma_tot,".--g", label = "Total uncertanty")
plt.legend()
plt.ylabel(r"$\sigma_{D}$ [Gy]")
plt.xlabel("Dose [Gy]")
plt.show()

#find uncertaty to old scan
mean_netOD = np.mean(np.reshape(scan1_netOD, (7,4)), axis = 1)
mean_sigma_netOD = np.mean(np.reshape(scan1_sigma_netOD, (7,4)), axis =
1)//np.sqrt(4)
uncert = uncert_Dtot(popt, mean_netOD, sigma_param, mean_sigma_netOD)
sigma_Dtot = interpolate.UnivariateSpline([0.1,0.2,0.5,1,2,5,10],uncert)
x = np.linspace(0.1,10,10000)
uncert_array = np.stack((x,sigma_Dtot(x)), axis = -1)
np.savetxt("/Users/bjorgvh/Dropbox/Master/protonuncertainty.txt",uncert_array)

plt.plot(x,sigma_Dtot(x), "-g", label = "Spline interpolation")
plt.plot([0.1,0.2,0.5,1,2,5,10], uncert, "og", label = "Mean total uncertainty per
dose")
plt.legend()
plt.ylabel(r"$\sigma_{\overline{Dose}}$ [Gy]")
plt.xlabel("Dose [Gy]")
plt.show()

dosepoint = np.repeat([0.1,0.2,0.5,1,2,5,10],2)
scan2_mean_exp1 = np.mean(scan2_dose[:,:2],axis=1)
scan2_mean_exp2 = np.mean(scan2_dose[:,2:],axis=1)
scan1_mean_exp1 = np.mean(model(np.reshape(scan1_exp1_netOD, (7,2)),popt),axis=1)
scan1_mean_exp2 = np.mean(model(np.reshape(scan1_exp2_netOD, (7,2)),popt),axis=1)
print np.mean((scan2_mean_exp1-scan1_mean_exp1,scan2_mean_exp2-
scan1_mean_exp2),axis=0)/np.mean(scan2_dose,axis=1) *100

plt.plot(dosepoint,scan2_exp1_netOD, ".k",  label= "true X-ray scan, experiment 1")
plt.plot(dosepoint,scan2_exp2_netOD, "^k", markersize =3.3,label = "true X-ray
scan, experiment 2")
plt.plot(dosepoint,scan1_exp1_netOD, ".", color="sandybrown",label="old X-ray scan,
experiment 1")
plt.plot(dosepoint,scan1_exp2_netOD, "^",color="sandybrown",markersize =3.3,  label
= "old X-ray scan, experiment 2")
plt.ylabel("netOD")
plt.xlabel("Dose [Gy]")
plt.legend()
plt.show()

plt.plot(dosepoint,scan2_exp1_dose,".k", label="true X-ray scan with original
calibraiton, experiment 1")
plt.plot(dosepoint,scan2_exp2_dose,"^k",markersize = 3.3 ,label="true X-ray scan
with original calibraiton, experiment 2")
plt.plot(dosepoint, model(scan1_exp1_netOD, popt), ".",color="sandybrown",
label="old X-ray scan with new calibration, experiment 1")
plt.plot(dosepoint, model(scan1_exp2_netOD, popt), "^",color="sandybrown",
markersize =3.3, label="old X-ray scan with new calibration, experiment 2")
plt.ylabel("Dose calculated from netOD [Gy]")
plt.xlabel("Nominal dose [Gy]")
plt.legend()
plt.show()
```

Proton films

```
# Analyse all proton films

import numpy as np
import matplotlib.pyplot as plt
import cv2
import glob
import os
from matplotlib.legend_handler import HandlerTuple

def dose_proton(roi,boolean = True): #model found from old x-ray films
    if boolean:
        i_exp = roi
    else:
        i_exp = np.mean(roi)
    i_unexp = 36047.50811682439
    i_bckg = 460.15472953428224
    netOD = np.log10((i_unexp-i_bckg)/(i_exp-i_bckg))
    return np.nan_to_num(31.47661638*netOD**1.25)

def dose_xray(roi): #best fit from orininal calibration
    i_exp = np.mean(roi)
    i_unexp = 29196.331339369095 #green channel
    i_bckg =  608.4299547416639 #green channel
    netOD = np.log10((i_unexp-i_bckg)/(i_exp-i_bckg))
    return np.nan_to_num(28.35820601*netOD**1.25)

def doseprofile(image):
    profile = np.zeros(image.shape[0])
    for i in range(image.shape[0]):
        profile[i] = np.sum(image[i])/np.sum(image[i]>0)
    return np.nan_to_num(profile)


def profile_per_dx(prof, wanted_dx):
    length = 306 # all images is 306 pixels long
    pixel_to_cm = 1./(150./2.54)  # 1 inch = 2.54 cm, 150 dpi image
    cm_to_pixel = 150./2.54
    dx_pixel = int(np.round(wanted_dx * cm_to_pixel)) #get dx in pixels

    n_bands = int(np.ceil(float(length)/dx_pixel))
    new_length = dx_pixel * n_bands
    new_prof = np.append(prof, np.zeros(new_length - length))

    reshaped = np.reshape(new_prof, (n_bands,-1)) #shape into bands of dx
    prof = np.mean(reshaped, axis =1)
    return prof, np.linspace(0,length*pixel_to_cm,n_bands)


def rotate_image(image, angle):
    image_center = tuple(np.array(image.shape[1::-1])/2)
    rot_mat = cv2.getRotationMatrix2D(image_center, angle, 1.0)
    rotated_image = cv2.warpAffine(image, rot_mat, image.shape[1::-1],
flags=cv2.INTER_LINEAR)
    return rotated_image


def phase_correlation(template, image):
    f1 = np.fft.fft2(template[50:225,50:225])
    results = np.zeros((361,2))
    angles = np.append(np.arange(10),np.arange(350,360))

    for angle in angles:
        rotated = rotate_image(image, angle)
        f0 = np.fft.fft2(rotated[50:225,50:225])
        correlation = np.abs(np.fft.ifft2( (f0 * np.conj(f1))/np.abs(f0*f1) ))

        results[angle,0] = np.unravel_index(np.argmax(correlation), f0.shape)[0]
        results[angle,1] = np.max(correlation)

    best_angle = np.argmax(results[:,1]) #highest number = best match
    shift = results[best_angle,0]

    if shift > f1.shape[0] // 2:
        shift -= f1.shape[0]

    best_rotation = rotate_image(image, best_angle)

    #shift image to match dose template :
    shifted_img = best_rotation
    if shift > 0:
        shift = int(shift)
        shifted_img = np.zeros((best_rotation.shape[0]+shift,
best_rotation.shape[1]), dtype = np.uint16)
        shifted_img[:-np.abs(shift),:] = best_rotation
        shifted_img = shifted_img[int(np.abs(shift)):,:]
    if shift < 0:
        shift = int(np.abs(shift))
        shifted_img = np.zeros((best_rotation.shape[0]+shift,
best_rotation.shape[1]), dtype = np.uint16)
        shifted_img[shift:,:] = best_rotation
        shifted_img = shifted_img[:-shift,:]

    return np.nan_to_num(shifted_img)



#get open field proton films
path_prot = "/Users/bjorgvh/Dropbox/Master/scan_film_proton_18okt"
filelist = glob.glob(os.path.join(path_prot, "*.tif"))
filelist.sort()
```

```
prot_dose1 = np.zeros(len(filelist))

for i in range(len(filelist)):
    img = cv2.imread(filelist[i], cv2.IMREAD_UNCHANGED)[:,:,1]
    min = img.shape[1]/2 - 24
    max = img.shape[1]/2 + 24
    prot_dose1[i] = dose_proton(img[min:max,min:max],False)

#get X-ray dcalibration films with nominal 2,5,10 Gy
path_xray = "/Users/bjorgvh/Dropbox/Master/kalibreringsfilmer"
filelist = glob.glob(os.path.join(path_xray, "*.tif"))
xrayfiles = []
for file in filelist[:]:
    if file[-15:-12] == "-2g" or file[-15:-12] == "-5g" or file[-15:-12] == "10g":
        xrayfiles.append(file)
xrayfiles.sort()

xray_dose = np.zeros(len(xrayfiles))
xray_netOD = np.zeros(len(xrayfiles))
for i in range(len(xray_dose)):
    img = cv2.imread(xrayfiles[i], cv2.IMREAD_UNCHANGED)[:,:,1]
    min = img.shape[1]/2 - 95
    max = img.shape[1]/2 + 95
    xray_dose[i] = dose_xray(img[min:max,min:max])


dosepoint_protondag1 = np.array([2.01,2.09,4.99,5.12,10.04,10.06]) #true doses
delivered
dosepoint_protondag2 = np.array([2.04,2.05,5.02,5.04,9.93,9.97]) #true doses
delivered
dosepoint_xray = np.repeat([2,5,10], 4)

#plot open filed with calibration films from X-ray
plt.plot(dosepoint_protondag1, prot_dose1[:6],".g",color = "lightskyblue",label =
"Proton experiment 1")
plt.plot(dosepoint_protondag2, prot_dose1[6:],".g",color = "steelblue", label =
"Proton experiment 2")
plt.plot(dosepoint_xray,xray_dose,".k", label="X-ray calibration films")
plt.ylabel("Dose calculated from netOD [Gy]")
plt.xlabel("Nominal dose [Gy]")
plt.legend()
plt.show()


#get GRID proton films
gridpath = "/Users/bjorgvh/Dropbox/Master/scan_film_proton_18okt/GRID_fix"
filelist = glob.glob(os.path.join(gridpath, "*.tif"))
filelist.sort()

#phase correlation, cropping and concert to dose
template = rotate_image(cv2.imread(filelist[1], cv2.IMREAD_UNCHANGED)[:,:,1],-0.5)
profiles = np.zeros((4,template.shape[0]))
```

```
dosemap_grid = np.zeros((4, template.shape[0],template.shape[1]))
for i in range(4):
    corr_grid = phase_correlation(template, cv2.imread(filelist[i],
cv2.IMREAD_UNCHANGED)[:,:,1])
    r = 140
    a1 = corr_grid.shape[0]/2
    b1 = corr_grid.shape[1]/2
    mask = np.full(corr_grid.shape, 0, dtype = np.uint16)
    cv2.circle(mask, (a1,b1), r, (65535,65535,65535), -1)
    crop = cv2.bitwise_and(corr_grid, mask)
    dosemap = dose_proton(crop)
    dosemap_grid[i] = dosemap
    profiles[i] = doseprofile(dosemap)

#plot dose maps of proton GRID films
fig, axs = plt.subplots(1,4)
ax1 = axs[0].imshow(dosemap_grid[0],vmin=0, vmax=6)
axs[1].imshow(dosemap_grid[1],vmin=0, vmax=6)
axs[2].imshow(dosemap_grid[2],vmin=0, vmax=6)
axs[3].imshow(dosemap_grid[3],vmin=0, vmax=6)
cbar_ax = fig.add_axes([0.95, 0.35, 0.01, 0.27])
plt.colorbar(ax1, cax=cbar_ax)
plt.show()

#get open field profiles, crop
dag1_5gy1 =
cv2.imread("/Users/bjorgvh/Dropbox/Master/scan_film_proton_18okt/open5_fix/dag1_05g
y3953-kopi.tif", cv2.IMREAD_UNCHANGED)[:,:,1]
dag1_5gy2 =
cv2.imread("/Users/bjorgvh/Dropbox/Master/scan_film_proton_18okt/open5_fix/dag1_05g
y4056-kopi.tif", cv2.IMREAD_UNCHANGED)[:,:,1]
dag1_open1 =cv2.bitwise_and(dag1_5gy1,mask)
dag1_open2 =cv2.bitwise_and(dag1_5gy2,mask)
dag2_5gy1 =
cv2.imread("/Users/bjorgvh/Dropbox/Master/scan_film_proton_18okt/open5_fix/dag2_05g
y3947-kopi.tif", cv2.IMREAD_UNCHANGED)[:,:,1]
dag2_5gy2 =
cv2.imread("/Users/bjorgvh/Dropbox/Master/scan_film_proton_18okt/open5_fix/dag2_05g
y3959-kopi.tif", cv2.IMREAD_UNCHANGED)[:,:,1]
dag2_open1 =cv2.bitwise_and(dag2_5gy1,mask)
dag2_open2 =cv2.bitwise_and(dag2_5gy2,mask)
openfield5gy = np.array([doseprofile(dose_proton(dag1_open1)),
doseprofile(dose_proton(dag1_open2)),doseprofile(dose_proton(dag2_open1)),doseprofi
le(dose_proton(dag2_open2))])

#plot open field vs GRID
x = np.linspace(0,template.shape[0]/(150/2.54),template.shape[0])
plt.plot(x,profiles[0],color="tan",label = "Experiment 1, GRID")
plt.plot(x,profiles[1],color="tan")
plt.plot(x,profiles[2],color="darkslategray",label ="Experiment 2, GRID")
plt.plot(x,profiles[3],color="darkslategray")
plt.plot(x[15:290],doseprofile(dose_proton(dag1_open1))[15:290], "--",color="tan",
label = "Experiment 1, open field")
plt.plot(x[15:290],doseprofile(dose_proton(dag1_open2))[15:290],"--",color="tan")
plt.plot(x[15:290],doseprofile(dose_proton(dag2_open1))[15:290],"--
",color="darkslategray", label =  "Experiment 2, open field")
plt.plot(x[15:290],doseprofile(dose_proton(dag2_open2))[15:290],"--
",color="darkslategray")
plt.ylabel("Dose [Gy]")
plt.xlabel("Position in petri dish [cm]")
plt.legend(loc="center left")
plt.show()


#find true 2,5,10 GY doseprofiles with 95%conf.bands
mean_profile = np.mean(profiles, axis = 0)
mean_profile2gy = 2./5*mean_profile
mean_profile10gy = 2*mean_profile
mean_open_field = np.mean(openfield5gy, axis = 0)

dosepoint= []; doseuncert =[]
with open("/Users/bjorgvh/Dropbox/Master/protonuncertainty.txt", "r") as file:
    for line in file:
        data = line.split()
        dosepoint.append(data[0]); doseuncert.append(data[1])
sigma_Dfit_list = np.stack((np.array(dosepoint, dtype = float),np.array(doseuncert,
dtype = float)), axis = -1)
sigma_Dfit_meanprofile = np.zeros(len(mean_profile))
sigma_Dfit_meanprofile2gy = np.zeros(len(mean_profile))
sigma_Dfit_meanprofile10gy = np.zeros(len(mean_profile))

sigma_Dfit_openfield = np.zeros(len(mean_open_field))

for i in range(len(mean_profile)):
    sigma_Dfit_meanprofile[i] =
sigma_Dfit_list[np.argmin(np.abs(sigma_Dfit_list[:,0]-mean_profile[i])), 1]
    sigma_Dfit_meanprofile2gy[i] =
sigma_Dfit_list[np.argmin(np.abs(sigma_Dfit_list[:,0]-mean_profile2gy[i])), 1]
    sigma_Dfit_meanprofile10gy[i] =
sigma_Dfit_list[np.argmin(np.abs(sigma_Dfit_list[:,0]-mean_profile10gy[i])), 1]
    sigma_Dfit_openfield[i] =
sigma_Dfit_list[np.argmin(np.abs(sigma_Dfit_list[:,0]-mean_open_field[i])), 1]

sigma_GRID = np.std(profiles, axis=0)
conf_band = 1.96*np.sqrt(sigma_Dfit_meanprofile**2 +
sigma_GRID**2/np.sqrt(profiles.shape[0]))
conf_band2gy = 1.96*np.sqrt(sigma_Dfit_meanprofile2gy**2 +
sigma_GRID**2/np.sqrt(profiles.shape[0]))
conf_band10gy = 1.96*np.sqrt(sigma_Dfit_meanprofile10gy**2 +
sigma_GRID**2/np.sqrt(profiles.shape[0]))

sigma_open = np.std(openfield5gy, axis=0)
```

```
conf_band_openfield = 1.96*np.sqrt(sigma_Dfit_openfield**2 +
sigma_open**2/np.sqrt(openfield5gy.shape[0]))
openf = np.mean(mean_open_field[20:275])

print "mean GRID 2gy ", np.mean(mean_profile2gy[20:275])
print "mean GRID 5gy ", np.mean(mean_profile[20:275])
print "mean GRID 10gy ", np.mean(mean_profile10gy[20:275])
print "mean open field 2gy ", 2./5*openf
print "mean open field 5gy ", openf,  np.mean(conf_band_openfield)
print "mean open field 10gy ", 2*openf

plt.plot(x[:280],mean_profile[:280], "k", label = "Mean profile, GRID")
plt.fill_between(x[:280], mean_profile[:280]-conf_band[:280],
mean_profile[:280]+conf_band[:280], color = "0.75")
plt.plot(x[15:280], mean_open_field[15:280], "--k", label = "Mean profile, open
field")
plt.fill_between(x[15:280], mean_open_field[15:280]-conf_band_openfield[15:280],
mean_open_field[15:280]+conf_band_openfield[15:280], color = "0.75", alpha = 0.5,
label = "95% confidence interval")
plt.ylabel("Dose [Gy]")
plt.xlabel("Position in flask [cm]")
plt.title("Mean dose profile for proton GRID films")
plt.legend(loc = "center left")
plt.show()

# Saving dose profiles per band width, same dx as when counting cells
mean_profile_dx05, x05_dose = profile_per_dx(mean_profile, 0.05)
mean_profile_dx15, x15_dose = profile_per_dx(mean_profile, 0.15)
conf_band_dx05,x1 = profile_per_dx(conf_band, 0.05)
conf_band_dx15,x1 = profile_per_dx(conf_band, 0.15)

mean_profile2gy_dx05, x05_dose = profile_per_dx(mean_profile2gy, 0.05)
mean_profile2gy_dx15, x15_dose = profile_per_dx(mean_profile2gy, 0.15)
conf_band2gy_dx05,x1 = profile_per_dx(conf_band2gy, 0.05)
conf_band2gy_dx15,x1 = profile_per_dx(conf_band2gy, 0.15)

mean_profile10gy_dx05, x05_dose = profile_per_dx(mean_profile10gy, 0.05)
mean_profile10gy_dx15, x15_dose = profile_per_dx(mean_profile10gy, 0.15)
conf_band10gy_dx05,x1 = profile_per_dx(conf_band10gy, 0.05)
conf_band10gy_dx15,x1 = profile_per_dx(conf_band10gy, 0.15)

plt.plot(mean_profile10gy_dx05)
plt.plot(mean_profile10gy_dx05-conf_band10gy_dx05, "--k")
plt.plot(mean_profile10gy_dx05+conf_band10gy_dx05, "--k")
plt.show()

np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose.txt", mean_profile)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx05.txt",
mean_profile_dx05)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx15.txt",
mean_profile_dx15)
```

```
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/conf_band.txt", conf_band)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx05.txt",
conf_band_dx05)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx15.txt",
conf_band_dx15)

np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose10gy.txt",
mean_profile10gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx05_10gy.txt",
mean_profile10gy_dx05)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx15_10gy.txt",
mean_profile10gy_dx15)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/conf_band10gy.txt",
conf_band10gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx05_10gy.txt",
conf_band10gy_dx05)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx15_10gy.txt",
conf_band10gy_dx15)

np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose2gy.txt",
mean_profile2gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx05_2gy.txt",
mean_profile2gy_dx05)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx15_2gy.txt",
mean_profile2gy_dx15)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/conf_band2gy.txt",
conf_band2gy)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx05_2gy.txt",
conf_band2gy_dx05)
np.savetxt("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx15_2gy.txt",
conf_band2gy_dx15)


# Compare mean dose profile from X-ray with proton

pixel_to_cm = 1./(1200./2.54)  # 1 inch = 2.54 cm, 1200 dpi image
cm = np.linspace(0,3000*pixel_to_cm,3000)#[500:2700]
meanXray = np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose.txt",
"r").readlines(), dtype = float)#[500:2700]
confXray = np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/conf_band.txt",
"r").readlines(), dtype = float)#[500:2700]
inter_xray = np.interp(x[:280], cm[:2200], meanXray[500:2700])
inter_confxray = np.interp(x[:280], cm[:2200], confXray[500:2700])

f1 = np.fft.fft(mean_profile[:280]); f0 = np.fft.fft(inter_xray)
correlation = np.abs(np.fft.ifft((f0 * np.conj(f1))))
shift = np.unravel_index(np.argmax(correlation), f0.shape)[0]
shift = np.abs(shift - f1.shape[0])

shifted = np.zeros(len(inter_xray) + shift)
shifted[shift:] = inter_xray
```

```
shifted_xray = shifted[:-shift]
shifted2 = np.zeros(len(inter_confxray) + shift)
shifted2[shift:] = inter_confxray
shifted_confxray = shifted2[:-shift]

prot, = plt.plot(x[15:280],mean_profile[15:280],color = "sienna")
xray, = plt.plot(x[15:280],shifted_xray[15:],"k")
ci_prot = plt.fill_between(x[15:280], mean_profile[15:280]-conf_band[15:280],
mean_profile[15:280]+conf_band[15:280], color = "peru", alpha = 0.5)
ci_xray = plt.fill_between(x[15:280], shifted_xray[15:]-shifted_confxray[15:],
shifted_xray[15:]+shifted_confxray[15:], color = "0.5", alpha = 0.5)
plt.legend([prot, xray, (ci_prot,ci_xray)], ["Mean profile proton GRID","Mean
profile X-ray GRID", "95% confidence intervals"], handler_map={tuple:
HandlerTuple(ndivide=None)},loc = (0,0.25))
plt.ylabel("Dose [Gy]")
plt.xlabel("[cm]")
plt.title("Mean dose profile for X-ray and proton GRID films")
plt.show()
plt.plot(mean_profile,color = "sienna")
plt.plot(shifted_xray,"k")
plt.show()
```

Compare X-ray dosimetry with Monte Carlo

```
# PLOT MC results against dose profile from EBT3 film dosimetry

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

def dose_MC(filename): #get data from monte carlo simulations

    #with open("/Users/bjorgvh/Dropbox/Master/MCdata" + filename) as csv_file:
    #    mc = list((csv.reader(csv_file)))
    #mc = np.array(mc, dtype = float)

    file = open("/Users/bjorgvh/Dropbox/Master/MCdata" + filename, "r").readlines()
    mc = np.zeros((401,401))
    for i in range(len(file)):
        mc[i] = np.array(file[i].split(","),dtype = float)

    mcprofile = np.zeros(mc.shape[0])
    for i in range(mc.shape[0]):
        mcprofile[i] = np.sum(mc[i])/mc.shape[1]

    return mcprofile

x = np.linspace(0, 3000./(1200./2.54), 3000) #3000 pixels, 1200dpi, 1 inch = 2.54
cm
mcx = np.linspace(0, 401.*0.01616, 401) # 301 pixels, spatial resuolution 0.0225 in
y-direction

mc_flaskA = dose_MC("/Dose_Flask4.txt")
mc_flaskB = dose_MC("/Dose_Flask3.txt")
mc_flaskC = dose_MC("/Dose_Flask2.txt")
mc_flaskD = dose_MC("/Dose_Flask1.txt")
mc_mean =
np.mean((mc_flaskA,mc_flaskB,np.flip(mc_flaskC,axis=0),np.flip(mc_flaskD,axis=0)),
axis=0)

mc_top = np.median((mc_mean[110:136],mc_mean[208:234],mc_mean[306:332]))
mc_valley = np.median((mc_mean[45:105],mc_mean[142:202],mc_mean[240:300]))
mc_mean /= mc_top

film_mean = np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose.txt",
"r").readlines(), dtype = float)
confband = np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/conf_band.txt",
"r").readlines(), dtype = float)
confband /=np.max(film_mean)
film_mean /= np.max(film_mean)

film_mean_interpolated = np.interp(mcx, x, film_mean)
confband_interpolated = np.interp(mcx, x, confband)
```

```
f1 = np.fft.fft(mc_mean); f0 = np.fft.fft(film_mean_interpolated)
correlation = np.abs(np.fft.ifft((f0 * np.conj(f1))))
shift = np.unravel_index(np.argmax(correlation), f0.shape)[0]
shift = np.abs(shift - f1.shape[0])

shifted_img = np.zeros(len(film_mean_interpolated) + shift)
shifted_img[shift:] = film_mean_interpolated
shifted_film = shifted_img[:-shift]
shifted_img2 = np.zeros(len(confband_interpolated) + shift)
shifted_img2[shift:] = confband_interpolated
shifted_conf= shifted_img2[:-shift]

film_top =
np.median((shifted_film[110:136],shifted_film[208:234],shifted_film[306:332]))
film_valley =
np.median((shifted_film[45:105],shifted_film[142:202],shifted_film[240:300]))
rel_diff = 100. * np.abs(shifted_film-mc_mean)/((shifted_film+mc_mean)/2.)

mc_val = np.array((mc_mean[45:105],mc_mean[142:202],mc_mean[240:300]))
film_val =
np.array((shifted_film[45:105],shifted_film[142:202],shifted_film[240:300]))
print "Peak/valley ratio gafchromc film: ", film_top/film_valley
print "Peak/valley ratio MC simulation : ", mc_top/mc_valley

plt.plot(mcx[45:-30],shifted_film[45:-30],"k", label = "EBT3 films")
plt.fill_between(mcx[45:-30], shifted_film[45:-30]-shifted_conf[45:-30],
shifted_film[45:-30]+shifted_conf[45:-30], color = "0.75", label = "95% CI EBT3
films")
plt.plot(mcx[45:-30],mc_mean[45:-30], "indianred", label = "Monte Carlo")
plt.legend(loc = "center left")
plt.ylabel("Dose [normalised]")
plt.title("GRID dose profile from EBT3 films vs Monte Carlo simulations")
plt.xlabel("Position in flask [cm]")
plt.show()
```

Cell quantification by share of colored pixels

```
# morfological opening, otsus thresholding. Find share of colored cells per pixel
row

import cv2
import matplotlib.pyplot as plt
import numpy  as np
from skimage.filters import threshold_otsu

img = cv2.imread("xray_crop.tif", cv2.IMREAD_UNCHANGED)[:,:,2]
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (10,10))
open = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
thresh = threshold_otsu(open[open>0])

cells = np.zeros(img.shape[0])
for i in range(img.shape[0]):
    shape = np.float64(np.count_nonzero(open[i]))
    cells[i] = ((0 < open[i]) & (open[i] < thresh)).sum() / shape
cells = np.nan_to_num(cells)

pixel_to_cm = 1./(1200./2.54)  # 1 inch = 2.54 cm, 1200 dpi image
cm = np.linspace(0,3000*pixel_to_cm,3000)

plt.imshow(img, cmap = "gray")
plt.show()
plt.imshow(open, cmap= "gray")
plt.show()
plt.imshow(open>thresh, cmap = "gray")
plt.show()

plt.plot(cm, cells*100, "k")
plt.ylabel("% of flask covered with cells")
plt.xlabel("Position in flask [cm]")
plt.show()
```

Get cell data from Arous' program

```
# use data from cell count (Arous,D.), find survival per dx.

import xlrd
import csv
import glob
import os
import numpy as np
import matplotlib.pyplot as plt
import codecs


def count_colonies(file, template, dx):

    with open(template, "rU") as csv_file: #extracting shape of flask from template
        temp = list((csv.reader(csv_file)))
    temp = np.array(temp, dtype = int)
    flask_shape = np.sum(temp, axis = 1, dtype = "float") #template consist of 1 
when flask, 0 when outside
    flask_shape /= np.max(flask_shape) #normalise so widest part of flask is 1.

    colonydata = xlrd.open_workbook(file) #open colonydata, excel file
    colsheet = colonydata.sheet_by_index(0)
    cols_centre = np.sort(colsheet.col_values(2)[1:]).astype("int") #centre of all 
colonies (y_value) by pixel value, sorted

    pixel_to_cm = 1./(1200./2.54)  #1200 dpi image, 1 inch = 2.54 cm
    length = len(flask_shape)

    #count all colonies located in each pixel column:
    cols_pr_pixel = np.zeros(length)
    for i in range(length):
        cols_pr_pixel[i] = len(cols_centre[cols_centre == i])
    cols_pr_pixel /= flask_shape #adjust for shape of flask

    #count colonies ber band width (dx)
    pixels_in_band = np.round(dx/pixel_to_cm)  #find number of pixels in band
    n_bands = int(np.ceil(length/pixels_in_band)) #number of bands in flask
    zeros = np.zeros(int(pixels_in_band*n_bands) -  length)
    cols_per_pixel_expanded = np.append(cols_pr_pixel, zeros) #add zeros to end to 
make it whole division of flask length (OBS makes last band different from rest)

    band_shaped = np.reshape(cols_per_pixel_expanded, (n_bands, -1)) #reshape into 
bands of thickness dx
    cols_pr_band = np.sum(band_shaped, axis = 1)/(30000./n_bands) #count colonies 
per band, divide by plated cells per band (30000 plated cells per flask)
    x_array = np.linspace(0,length*pixel_to_cm,n_bands) #x in cm

    return  cols_pr_band, x_array
```

```
def read_all_to_file(path, dx, xray = True):

    tempfile = glob.glob(os.path.join(path, "*.csv"))
    datafile =  glob.glob(os.path.join(path, "*.xlsx"))

    templist = []
    datalist = []
    for file in sorted(tempfile):
        templist.append(file)
    for file in sorted(datafile):
        datalist.append(file)

    count = []
    if xray:
        for i in range(len(templist)):
            colonycount, x = count_colonies(datalist[i],templist[i],dx)
            filename =  datalist[i][30:-16].replace("/", "-") + "-dx" + str(dx)[-
2:]
            #np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/%s.txt"
%filename, colonycount)
            count.append(colonycount)
        filename = "xarray" + str(dx)[-2:]
        np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/%s.txt"
%filename, x)

    else:
        for i in range(len(templist)):
            colonycount, x = count_colonies(datalist[i],templist[i],dx)
            filename =  datalist[i][30:-16].replace("/", "-") + "-dx" + str(dx)[-
2:]
            #np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/%s.txt"
%filename, colonycount)
            count.append(colonycount)
        filename = "xarray" + str(dx)[-2:]
        np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/%s.txt"
%filename, x)
    return np.array(count)




    # TEST TO DECIDE DX:
testfile = "/Users/bjorgvh/Dropbox/Master/20112019/GRID_Stripes/A549-2011-05-gridS-
A-ColonyData.xlsx"
testtemp = "/Users/bjorgvh/Dropbox/Master/20112019/GRID_Stripes/A549-2011-05-gridS-
A-Template.csv"


count_dx025, x025 = count_colonies(testfile, testtemp, 0.025)

count_dx05, x05 = count_colonies(testfile, testtemp, 0.05)
count_dx1, x1 = count_colonies(testfile, testtemp, 0.1)
```

```
count_dx15, x15 = count_colonies(testfile, testtemp, 0.15)


fig, axs = plt.subplots(4, sharex=True, sharey=True, gridspec_kw={'hspace': 0})
axs[0].plot(x025, count_dx025,  label = "dx = 0.25 mm", color = "steelblue")
axs[1].plot(x05, count_dx05, label = "dx = 0.5 mm", color = "steelblue")
axs[2].plot(x1, count_dx1, label = "dx = 1.0 mm", color = "steelblue")
axs[3].plot(x15, count_dx15, label = "dx = 1.5 mm", color = "steelblue")
axs[0].legend(loc = "lower right")
axs[1].legend(loc = "lower right")
axs[2].legend(loc = "lower right")
axs[3].legend(loc = "lower right")
fig.text(0.02, 0.6, "Plating efficiency",  rotation="vertical")
plt.xlabel("Position in flask [cm]")
plt.show()



    #PROTON DATA:
path_protoncontrol_tue = "/Users/bjorgvh/Dropbox/Master/ProtonTuesday2019/Control"
path_protoncontrol_thur =
"/Users/bjorgvh/Dropbox/Master/ProtonThursday2019/Control"
path_proton_tue = "/Users/bjorgvh/Dropbox/Master/ProtonTuesday2019/GRID Stripes"
path_proton_thur = "/Users/bjorgvh/Dropbox/Master/ProtonThursday2019/GRID Stripes"

control_prot_tue05 = read_all_to_file(path_protoncontrol_tue, 0.05, xray=False)
control_prot_thur05 = read_all_to_file(path_protoncontrol_thur, 0.05, xray=False)

prot_tue05 = read_all_to_file(path_proton_tue, 0.05, xray=False)
prot_thur05 = read_all_to_file(path_proton_thur, 0.05, xray=False)

prot_2gy_dx05 = np.concatenate((prot_tue05[:3]/np.mean(control_prot_tue05, axis =
0), prot_thur05[:3]/np.mean(control_prot_thur05,axis=0)),axis =0)
prot_5gy_dx05 = np.concatenate((prot_tue05[3:6]/np.mean(control_prot_tue05, axis =
0), prot_thur05[3:6]/np.mean(control_prot_thur05,axis=0)),axis =0)
prot_10gy_dx05 = np.concatenate((prot_tue05[6:9]/np.mean(control_prot_tue05, axis =
0), prot_thur05[6:9]/np.mean(control_prot_thur05,axis=0)),axis =0)
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean2Gy-dx05.txt",
np.nan_to_num(np.mean(prot_2gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean5Gy-dx05.txt",
np.nan_to_num(np.mean(prot_5gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean10Gy-dx05.txt",
np.nan_to_num(np.mean(prot_10gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std2Gy-dx05.txt",
np.nan_to_num(np.std(prot_2gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std5Gy-dx05.txt",
np.nan_to_num(np.std(prot_5gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std10Gy-dx05.txt",
np.nan_to_num(np.std(prot_10gy_dx05, axis = 0)))

control_prot_tue15 = read_all_to_file(path_protoncontrol_tue, 0.15, xray=False)
```

```
control_prot_thur15 = read_all_to_file(path_protoncontrol_thur, 0.15, xray=False)
prot_tue15 = read_all_to_file(path_proton_tue, 0.15, xray=False)
prot_thur15 = read_all_to_file(path_proton_thur, 0.15, xray=False)

prot_2gy_dx15 = np.concatenate((prot_tue15[:3]/np.mean(control_prot_tue15, axis =
0), prot_thur15[:3]/np.mean(control_prot_thur15,axis=0)),axis =0)
prot_5gy_dx15 = np.concatenate((prot_tue15[3:6]/np.mean(control_prot_tue15, axis =
0), prot_thur15[3:6]/np.mean(control_prot_thur15,axis=0)),axis =0)
prot_10gy_dx15 = np.concatenate((prot_tue15[6:9]/np.mean(control_prot_tue15, axis =
0), prot_thur15[6:9]/np.mean(control_prot_thur15,axis=0)),axis =0)
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean2Gy-dx15.txt",
np.nan_to_num(np.mean(prot_2gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean5Gy-dx15.txt",
np.nan_to_num(np.mean(prot_5gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean10Gy-dx15.txt",
np.nan_to_num(np.mean(prot_10gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std2Gy-dx15.txt",
np.nan_to_num(np.std(prot_2gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std5Gy-dx15.txt",
np.nan_to_num(np.std(prot_5gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std10Gy-dx15.txt",
np.nan_to_num(np.std(prot_10gy_dx15, axis = 0)))




    #X-RAY DATA:
path_contr18 = "/Users/bjorgvh/Dropbox/Master/18112019/Control"
path_contr20 = "/Users/bjorgvh/Dropbox/Master/20112019/Control"
path_open18 = "/Users/bjorgvh/Dropbox/Master/18112019/Open"
path_open20 = "/Users/bjorgvh/Dropbox/Master/20112019/Open"
path_grid18 = "/Users/bjorgvh/Dropbox/Master/18112019/GRID_Stripes"
path_grid20 = "/Users/bjorgvh/Dropbox/Master/20112019/GRID_Stripes"

control18_dx05 = read_all_to_file(path_contr18, 0.05)
control20_dx05 = read_all_to_file(path_contr20, 0.05)

open18_dx05 = read_all_to_file(path_open18, 0.05)
open20_dx05 = read_all_to_file(path_open20, 0.05)

open_2gy_dx05 = np.concatenate((open18_dx05[:4]/np.mean(control18_dx05[4:],axis=0),
open20_dx05[:4]/np.mean(control20_dx05,axis=0)), axis=0)
open_5gy_dx05 =
np.concatenate((open18_dx05[16:20]/np.mean(control18_dx05[4:],axis=0),
open20_dx05[4:8]/np.mean(control20_dx05,axis=0)), axis=0)
open_10gy_dx05 =
np.concatenate((open18_dx05[20:]/np.mean(control18_dx05[4:],axis=0),
open20_dx05[8:]/np.mean(control20_dx05,axis=0)), axis=0)
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_open2Gy-dx05.txt",
np.nan_to_num(np.mean(open_2gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_open5Gy-dx05.txt",
np.nan_to_num(np.mean(open_5gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_open10Gy-dx05.txt",
np.nan_to_num(np.mean(open_10gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_open2Gy-dx05.txt",
np.nan_to_num(np.std(open_2gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_open5Gy-dx05.txt",
np.nan_to_num(np.std(open_5gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_open10Gy-dx05.txt",
np.nan_to_num(np.std(open_10gy_dx05, axis = 0)))

grid18_dx05 = read_all_to_file(path_grid18, 0.05)
grid20_dx05 = read_all_to_file(path_grid20, 0.05)

grid_2gy_dx05 = np.concatenate((grid18_dx05[:4]/np.mean(control18_dx05[4:],axis=0),
grid20_dx05[:4]/np.mean(control20_dx05,axis=0)), axis=0)
grid_5gy_dx05 =
np.concatenate((grid18_dx05[16:20]/np.mean(control18_dx05[4:],axis=0),
grid20_dx05[4:8]/np.mean(control20_dx05,axis=0)), axis=0)
grid_10gy_dx05 =
np.concatenate((grid18_dx05[20:]/np.mean(control18_dx05[4:],axis=0),
grid20_dx05[8:]/np.mean(control20_dx05,axis=0)), axis=0)
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid2Gy-dx05.txt",
np.nan_to_num(np.mean(grid_2gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid5Gy-dx05.txt",
np.nan_to_num(np.mean(grid_5gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid10Gy-dx05.txt",
np.nan_to_num(np.mean(grid_10gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid2Gy-dx05.txt",
np.nan_to_num(np.std(grid_2gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid5Gy-dx05.txt",
np.nan_to_num(np.std(grid_5gy_dx05, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid10Gy-dx05.txt",
np.nan_to_num(np.std(grid_10gy_dx05, axis = 0)))

control18_dx15 = read_all_to_file(path_contr18, 0.15)
control20_dx15 = read_all_to_file(path_contr20, 0.15)

open18_dx15 = read_all_to_file(path_open18, 0.15)
open20_dx15 = read_all_to_file(path_open20, 0.15)

open_2gy_dx15 = np.concatenate((open18_dx15[:4]/np.mean(control18_dx15[4:],axis=0),
open20_dx15[:4]/np.mean(control20_dx15,axis=0)), axis=0)
open_5gy_dx15 =
np.concatenate((open18_dx15[16:20]/np.mean(control18_dx15[4:],axis=0),
open20_dx15[4:8]/np.mean(control20_dx15,axis=0)), axis=0)
open_10gy_dx15 =
np.concatenate((open18_dx15[20:]/np.mean(control18_dx15[4:],axis=0),
open20_dx15[8:]/np.mean(control20_dx15,axis=0)), axis=0)
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_open2Gy-dx15.txt",
np.nan_to_num(np.mean(open_2gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_open5Gy-dx15.txt",
np.nan_to_num(np.mean(open_5gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_open10Gy-dx15.txt",
np.nan_to_num(np.mean(open_10gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_open2Gy-dx15.txt",
np.nan_to_num(np.std(open_2gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_open5Gy-dx15.txt",
np.nan_to_num(np.std(open_5gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_open10Gy-dx15.txt",
np.nan_to_num(np.std(open_10gy_dx15, axis = 0)))

grid18_dx15 = read_all_to_file(path_grid18, 0.15)
grid20_dx15 = read_all_to_file(path_grid20, 0.15)

grid_2gy_dx15 = np.concatenate((grid18_dx15[:4]/np.mean(control18_dx15[4:],axis=0),
grid20_dx15[:4]/np.mean(control20_dx15,axis=0)), axis=0)
grid_5gy_dx15 =
np.concatenate((grid18_dx15[16:20]/np.mean(control18_dx15[4:],axis=0),
grid20_dx15[4:8]/np.mean(control20_dx15,axis=0)), axis=0)
grid_10gy_dx15 =
np.concatenate((grid18_dx15[20:]/np.mean(control18_dx15[4:],axis=0),
grid20_dx15[8:]/np.mean(control20_dx15,axis=0)), axis=0)
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid2Gy-dx15.txt",
np.nan_to_num(np.mean(grid_2gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid5Gy-dx15.txt",
np.nan_to_num(np.mean(grid_5gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid10Gy-dx15.txt",
np.nan_to_num(np.mean(grid_10gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid2Gy-dx15.txt",
np.nan_to_num(np.std(grid_2gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid5Gy-dx15.txt",
np.nan_to_num(np.std(grid_5gy_dx15, axis = 0)))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid10Gy-dx15.txt",
np.nan_to_num(np.std(grid_10gy_dx15, axis = 0)))

frac_control_dx05 = np.mean(control18_dx05[:4],axis=0)
frac_control_dx15 = np.mean(control18_dx15[:4],axis=0)

np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac1-grid-dx05.txt",
np.mean(grid18_dx05[4:6]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac2-grid-dx05.txt",
np.mean(grid18_dx05[6:8]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac3-grid-dx05.txt",
np.mean(grid18_dx05[8:10]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac4-grid-dx05.txt",
np.mean(grid18_dx05[10:12]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac5-grid-dx05.txt",
np.mean(grid18_dx05[12:14]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac6-grid-dx05.txt",
np.mean(grid18_dx05[14:16]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac1-open-dx05.txt",
np.mean(open18_dx05[4:6]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac2-open-dx05.txt",
np.mean(open18_dx05[6:8]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac3-open-dx05.txt",
np.mean(open18_dx05[8:10]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac4-open-dx05.txt",
np.mean(open18_dx05[10:12]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac5-open-dx05.txt",
np.mean(open18_dx05[12:14]/frac_control_dx05, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac6-open-dx05.txt",
np.mean(open18_dx05[14:16]/frac_control_dx05, axis = 0))

np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac1-grid-dx15.txt",
np.mean(grid18_dx15[4:6]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac2-grid-dx15.txt",
np.mean(grid18_dx15[6:8]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac3-grid-dx15.txt",
np.mean(grid18_dx15[8:10]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac4-grid-dx15.txt",
np.mean(grid18_dx15[10:12]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac5-grid-dx15.txt",
np.mean(grid18_dx15[12:14]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac6-grid-dx15.txt",
np.mean(grid18_dx15[14:16]/frac_control_dx15, axis = 0))

np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac1-open-dx15.txt",
np.mean(open18_dx15[4:6]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac2-open-dx15.txt",
np.mean(open18_dx15[6:8]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac3-open-dx15.txt",
np.mean(open18_dx15[8:10]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac4-open-dx15.txt",
np.mean(open18_dx15[10:12]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac5-open-dx15.txt",
np.mean(open18_dx15[12:14]/frac_control_dx15, axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/frac6-open-dx15.txt",
np.mean(open18_dx15[14:16]/frac_control_dx15, axis = 0))
```

Make example plot from GRID cells

```
#exampleplots of cell profiles

import numpy as np
import matplotlib.pyplot as plt
import glob
import os

x05 = np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/xarray05.txt",
"r").readlines(), dtype = float)
x15 = np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/xarray15.txt",
"r").readlines(), dtype = float)

grid18dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/18112019-
GRID_Stripes-A549-1811-05-gridS-A-dx05.txt", "r").readlines(), dtype = float)
grid18dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/18112019-
GRID_Stripes-A549-1811-05-gridS-A-dx15.txt", "r").readlines(), dtype = float)
grid20dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/20112019-
GRID_Stripes-A549-2011-05-gridS-A-dx05.txt", "r").readlines(), dtype = float)
grid20dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/20112019-
GRID_Stripes-A549-2011-05-gridS-A-dx15.txt", "r").readlines(), dtype = float)

mean18dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/meancontrol18-
dx05.txt", "r").readlines(), dtype = float)
mean18dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/meancontrol18-
dx15.txt", "r").readlines(), dtype = float)
mean20dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/meancontrol20-
dx05.txt", "r").readlines(), dtype = float)
mean20dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/meancontrol20-
dx15.txt", "r").readlines(), dtype = float)

open18dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/18112019-Open-A549-
1811-05-open-A-dx05.txt", "r").readlines(), dtype = float)
open18dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/18112019-Open-A549-
1811-05-open-A-dx15.txt", "r").readlines(), dtype = float)
open20dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/20112019-Open-A549-
2011-05-open-A-dx05.txt", "r").readlines(), dtype = float)
open20dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/20112019-Open-A549-
2011-05-open-A-dx15.txt", "r").readlines(), dtype = float)
```

```
plt.plot(x15,mean18dx15, ":", color = "grey",  label = "control")
plt.plot(x15,open18dx15,"--", color = "grey", label = "open field")
plt.plot(x15,grid18dx15,"-", color = "firebrick", label = "GRID")

plt.ylabel("Plating efficiency")
plt.title("GRID vs control vs open field, 5 Gy - dx = 1.5 mm")
plt.xlabel("Position in flask [cm]")
plt.ylim([0,0.25])
plt.legend()
plt.show()

plt.plot(x05,mean18dx05, ":", color = "grey",  label = "control")
plt.plot(x05,open18dx05,"--", color = "grey", label = "open field")
plt.plot(x05,grid18dx05,"-", color = "firebrick", label = "GRID")
plt.ylabel("Plating efficiency")
plt.xlabel("Position in flask [cm]")
plt.ylim([0,0.25])
plt.legend()
plt.show()

plt.plot(x05,grid18dx05, "-", color = "firebrick", label = "GRID, exp 1")
plt.plot(x05,grid20dx05, "-", color = "orange", label = "GRID, exp 2")
plt.plot(x05,mean18dx05, ":", color = "firebrick", label = "control, exp 1")
plt.plot(x05,mean20dx05, ":", color = "orange",  label = "control, exp 2")
plt.ylabel("Plating efficiency")
plt.xlabel("Position in flask [cm]")
plt.ylim([0,0.25])
plt.legend(ncol = 2)
plt.show()

plt.plot(x15,grid18dx15,"-", color = "firebrick", label = "GRID, exp 1")
plt.plot(x15,grid20dx15,"-", color = "orange", label = "GRID, exp 2")
plt.plot(x15,mean18dx15, ":", color = "firebrick",  label = "control, exp 1")
plt.plot(x15,mean20dx15,":", color = "orange",  label = "control, exp 2")
plt.ylabel("Plating efficiency")
plt.xlabel("Position in flask [cm]")
plt.ylim([0,0.25])
plt.legend(ncol=2)
plt.show()

plt.axhline(y=1., linestyle = "--", color = "gray")
plt.plot(x05,np.nan_to_num(grid18dx05/mean18dx05), "-", color = "firebrick",  label
= "exp 1")
plt.plot(x05,np.nan_to_num(grid20dx05/mean20dx05),"-", color = "orange", label =
"exp 2")
plt.ylabel("Survival fraction")
plt.xlabel("Position in flask [cm]")
plt.ylim([0,1.5])
plt.legend()
plt.show()
plt.axhline(y=1., linestyle= "--", color = "gray")
plt.plot(x15,np.nan_to_num(grid18dx15/mean18dx15),"-", color = "firebrick", label =
"exp 1")
plt.plot(x15,np.nan_to_num(grid20dx15/mean20dx15), "-", color = "orange", label =
"exp 2")
plt.ylabel("Survival fraction")
plt.xlabel("Position in flask [cm]")
plt.ylim([0,1.5])
plt.legend()
plt.show()
```

LQ-model from open field data

```python
#fitting LQ-model to data from open field irradaition of cells

import numpy as np
import matplotlib.pyplot as plt
import glob
import os
import xlrd
from scipy.optimize import curve_fit
from scipy import stats
import numpy.linalg
from kapteyn import kmpfit
from matplotlib.legend_handler import HandlerTuple

def colcount(path):
    colonies = []
    filelist = glob.glob(os.path.join(path, "*.xlsx"))
    for file in sorted(filelist):
        if len(file) == 75 or len(file) == 79:
            wb = xlrd.open_workbook(file)
            sheet = wb.sheet_by_index(0)
            colonies.append(sheet.nrows - 1)
    return np.array(colonies)

def log_lin(p, dose):
    alpha, beta = p
    return -alpha*dose - beta*dose**2


control18 = np.mean(colcount("/Users/bjorgvh/Dropbox/Master/18112019/Control"))
control20 = np.mean(colcount("/Users/bjorgvh/Dropbox/Master/20112019/Control"))
open18 =
np.reshape(colcount("/Users/bjorgvh/Dropbox/Master/18112019/Open")/control18,
(3,4))
open20 =
np.reshape(colcount("/Users/bjorgvh/Dropbox/Master/20112019/Open")/control20,
(3,4))
survival = np.ravel(np.reshape(np.stack((open18,open20), axis = -1), (3,8)))

true_5Gy = 4.588195 #mean of dose profile from open fields in flask
dose = np.repeat([true_5Gy*2./5, true_5Gy, true_5Gy*2], 8)
print dose[::8]

#do fit through kmp_fit, same results as curve_fit, but also provides confidence
band
x = np.linspace(0,10,10000)
f = kmpfit.simplefit(log_lin, [1, 1], dose, np.log(survival))
alpha1, beta1 = f.params
dfdp = [-x-beta1*x**2, -alpha1*x - x**2]
yhat, upper, lower = f.confidence_band(x, dfdp, 0.95, log_lin)
param = [alpha1, beta1]
```

```
print param, f.stderr

#get p-values from statsmodels
import statsmodels.api as sm
X =  np.stack((dose, dose**2), axis = -1)
y = -np.log(survival)
model = sm.OLS(y, X).fit()
predictions = model.predict(X)
print model.pvalues

#PLOT
fit, = plt.semilogy(x, np.exp(yhat), "-k")
confidence = plt.fill_between(x, np.exp(lower), np.exp(upper) , color = "0.75")
data1, = plt.semilogy(np.repeat([true_5Gy*2./5, true_5Gy, true_5Gy*2], 4),
np.ravel(open18), ".", color = "firebrick")
data2, = plt.semilogy(np.repeat([true_5Gy*2./5, true_5Gy, true_5Gy*2], 4),
np.ravel(open20), ".", color = "orange")
plt.legend([fit, confidence, (data1,data2)],[r"$-\alpha D - \beta D^2$","95%
confidence band","data points"], handler_map={tuple: HandlerTuple(ndivide=None)})
plt.xlabel("Dose [Gy]")
plt.ylabel("log(SF)")
plt.show()

np.savetxt("/Users/bjorgvh/Dropbox/Master/upper_conf.txt", upper)
np.savetxt("/Users/bjorgvh/Dropbox/Master/lower_conf.txt", lower)
np.savetxt("/Users/bjorgvh/Dropbox/Master/lqcurve.txt", yhat)
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_open18.txt",
np.mean(open18, axis =1))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_open20.txt",
np.mean(open20, axis =1))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/meanSF_openfield.txt",
np.mean((np.mean(open18, axis =1),np.mean(open20, axis =1)),axis = 0))
np.savetxt("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/meanSTD_openfield.txt",
np.std(np.reshape(np.stack((open18,open20), axis = -1), (3,8)), axis = 1))
```

Predicted and observed cell survival X-ray

```
#finding and plotting the pedicted and observed cell survival for X-ray GRID:

import glob
import os
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.legend_handler import HandlerTuple
from scipy.signal import argrelextrema

def log_sf(dose):
    alpha, beta = [0.07441434687842306, 0.025813497191013064]
    return -alpha*dose - beta*dose**2

sf_2gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid2Gy-
dx05.txt", "r").readlines(), dtype = float)
sf_5gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid5Gy-
dx05.txt", "r").readlines(), dtype = float)
sf_10gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid10Gy-
dx05.txt", "r").readlines(), dtype = float)
cell_conf_2gy_dx05 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid2Gy-
dx05.txt", "r").readlines(), dtype = float)/np.sqrt(8)
cell_conf_5gy_dx05 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid5Gy-
dx05.txt", "r").readlines(), dtype = float)/np.sqrt(8)
cell_conf_10gy_dx05 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid10Gy-
dx05.txt", "r").readlines(), dtype = float)/np.sqrt(8)

sf_2gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid2Gy-
dx15.txt", "r").readlines(), dtype = float)
sf_5gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid5Gy-
dx15.txt", "r").readlines(), dtype = float)
sf_10gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/mean_grid10Gy-
dx15.txt", "r").readlines(), dtype = float)
cell_conf_2gy_dx15 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid2Gy-
dx15.txt", "r").readlines(), dtype = float)/np.sqrt(8)
cell_conf_5gy_dx15 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid5Gy-
dx15.txt", "r").readlines(), dtype = float)/np.sqrt(8)
cell_conf_10gy_dx15 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/std_grid10Gy-
dx15.txt", "r").readlines(), dtype = float)/np.sqrt(8)
```

```
x05 = np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/xarray05.txt",
"r").readlines(), dtype = float)
x15 = np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/xarray15.txt",
"r").readlines(), dtype = float)

dose5gy = np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose.txt",
"r").readlines(), dtype = float)
dose5gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx05.txt",
"r").readlines(), dtype = float)
dose5gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx15.txt",
"r").readlines(), dtype = float)
dose5gy_conf_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx05.txt",
"r").readlines(), dtype = float)
dose5gy_conf_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx15.txt",
"r").readlines(), dtype = float)

dose2gy = np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose2gy.txt",
"r").readlines(), dtype = float)
dose2gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx05_2gy.txt",
"r").readlines(), dtype = float)
dose2gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx15_2gy.txt",
"r").readlines(), dtype = float)
dose2gy_conf_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx05_2gy.txt",
"r").readlines(), dtype = float)
dose2gy_conf_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx15_2gy.txt",
"r").readlines(), dtype = float)

dose10gy =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose10gy.txt",
"r").readlines(), dtype = float)
dose10gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx05_10gy.txt",
"r").readlines(), dtype = float)
dose10gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/meandose_dx15_10gy.txt",
"r").readlines(), dtype = float)
dose10gy_conf_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx05_10gy.txt",
"r").readlines(), dtype = float)
dose10gy_conf_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_xray/confband_dx15_10gy.txt",
"r").readlines(), dtype = float)
```

```
sf_upper_conf = np.array(open("/Users/bjorgvh/Dropbox/Master/upper_conf.txt",
"r").readlines(), dtype = float)
sf_lower_conf = np.array(open("/Users/bjorgvh/Dropbox/Master/lower_conf.txt",
"r").readlines(), dtype = float)
lqcurve = np.array(open("/Users/bjorgvh/Dropbox/Master/lqcurve.txt",
"r").readlines(), dtype = float)

expected_2gy_dx05 = np.exp(log_sf(dose2gy_dx05))
expected_5gy_dx05 = np.exp(log_sf(dose5gy_dx05))
expected_10gy_dx05 = np.exp(log_sf(dose10gy_dx05))

expected_2gy_dx15 = np.exp(log_sf(dose2gy_dx15))
expected_5gy_dx15 = np.exp(log_sf(dose5gy_dx15))
expected_10gy_dx15 = np.exp(log_sf(dose10gy_dx15))

sf_conf_low_2gy_dx05 = np.zeros(dose2gy_dx05.shape)
sf_conf_high_2gy_dx05 = np.zeros(dose2gy_dx05.shape)
for i in range(dose2gy_dx05.shape[0]):
    sf_conf_high_2gy_dx05[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose2gy_dx05[i]-dose2gy_conf_dx05[i]))).argmin()])
    sf_conf_low_2gy_dx05[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose2gy_dx05[i]+dose2gy_conf_dx05[i]))).argmin()])

sf_conf_low_5gy_dx05 = np.zeros(dose5gy_dx05.shape)
sf_conf_high_5gy_dx05 = np.zeros(dose5gy_dx05.shape)
for i in range(dose5gy_dx05.shape[0]):
    sf_conf_high_5gy_dx05[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose5gy_dx05[i]-dose5gy_conf_dx05[i]))).argmin()])
    sf_conf_low_5gy_dx05[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose5gy_dx05[i]+dose5gy_conf_dx05[i]))).argmin()])

sf_conf_low_10gy_dx05 = np.zeros(dose10gy_dx05.shape)
sf_conf_high_10gy_dx05 = np.zeros(dose10gy_dx05.shape)
for i in range(dose10gy_dx05.shape[0]):
    sf_conf_high_10gy_dx05[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose10gy_dx05[i]-dose10gy_conf_dx05[i]))).argmin()])
    sf_conf_low_10gy_dx05[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose10gy_dx05[i]+dose10gy_conf_dx05[i]))).argmin()])

sf_conf_low_2gy_dx15 = np.zeros(dose2gy_dx15.shape)
sf_conf_high_2gy_dx15 = np.zeros(dose2gy_dx15.shape)
for i in range(dose2gy_dx15.shape[0]):
    sf_conf_high_2gy_dx15[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose2gy_dx15[i]-dose2gy_conf_dx15[i]))).argmin()])
    sf_conf_low_2gy_dx15[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose2gy_dx15[i]+dose2gy_conf_dx15[i]))).argmin()])

sf_conf_low_5gy_dx15 = np.zeros(dose5gy_dx15.shape)
sf_conf_high_5gy_dx15 = np.zeros(dose5gy_dx15.shape)
for i in range(dose5gy_dx15.shape[0]):
    sf_conf_high_5gy_dx15[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose5gy_dx15[i]-dose5gy_conf_dx15[i]))).argmin()])
    sf_conf_low_5gy_dx15[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose5gy_dx15[i]+dose5gy_conf_dx15[i]))).argmin()])

sf_conf_low_10gy_dx15 = np.zeros(dose10gy_dx15.shape)
sf_conf_high_10gy_dx15 = np.zeros(dose10gy_dx15.shape)
for i in range(dose10gy_dx15.shape[0]):
    sf_conf_high_10gy_dx15[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose10gy_dx15[i]-dose10gy_conf_dx15[i]))).argmin()])
    sf_conf_low_10gy_dx15[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose10gy_dx15[i]+dose10gy_conf_dx15[i]))).argmin()])




# AVERAGE DOSE STUFF
meandose_2gy = np.mean(dose2gy[300:2800])
meandose_5gy = np.mean(dose5gy[300:2800])
meandose_10gy = np.mean(dose10gy[300:2800])

print meandose_2gy, meandose_5gy, meandose_10gy
openfieldSF =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/meanSF_openfield.txt"
, "r").readlines(), dtype = float)
openfieldSTD =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_xray/meanSTD_openfield.txt
", "r").readlines(), dtype = float)

lowerr_2gy = np.exp(log_sf(meandose_2gy))- np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(meandose_2gy+0.18604486230559028))).argmin()])
higherr_2gy = np.exp(sf_upper_conf[(np.abs(lqcurve - log_sf(meandose_2gy-
0.18604486230559028))).argmin()]) -np.exp(log_sf(meandose_2gy))
lowerr_5gy = np.exp(log_sf(meandose_5gy))- np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(meandose_5gy+0.3578308755643193))).argmin()])
higherr_5gy = np.exp(sf_upper_conf[(np.abs(lqcurve - log_sf(meandose_5gy-
0.3578308755643193))).argmin()]) -np.exp(log_sf(meandose_5gy))
lowerr_10gy = np.exp(log_sf(meandose_10gy))- np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(meandose_10gy+0.6516344729461035))).argmin()])
higherr_10gy = np.exp(sf_upper_conf[(np.abs(lqcurve - log_sf(meandose_10gy-
0.6516344729461035))).argmin()]) -np.exp(log_sf(meandose_10gy))
plt.errorbar(np.zeros(1), np.zeros([1,3]), color="w", alpha=0, label=" ") #dummy
legend

#open field
p7, caps7, bars7 = plt.errorbar(1.8352, openfieldSF[0], yerr = openfieldSTD[0],
marker = "o",markersize = 3.3, color = "saddlebrown", linestyle = "None", capsize =
3, label = "X-ray open field 2gy")
p8, caps8, bars8 = plt.errorbar(4.5882, openfieldSF[1], yerr = openfieldSTD[1],
marker = "^",markersize = 3.3, color = "saddlebrown", linestyle = "None", capsize =
3, label = "X-ray open field 5gy")
```

```
p9, caps9, bars9 = plt.errorbar(9.1764, openfieldSF[2], yerr = openfieldSTD[2],
marker = "s",markersize = 3.3, color = "saddlebrown", linestyle = "None", capsize =
3, label = "X-ray open field 10gy")
[bar.set_alpha(0.5) for bar in bars7];[cap.set_alpha(0.5) for cap in caps7]
[bar.set_alpha(0.5) for bar in bars8];[cap.set_alpha(0.5) for cap in caps8]
[bar.set_alpha(0.5) for bar in bars9];[cap.set_alpha(0.5) for cap in caps9]

#X-ray GRID
p4, caps4, bars4 = plt.errorbar(meandose_2gy, np.mean(sf_2gy_dx05), yerr =
np.mean(cell_conf_2gy_dx05), marker = "o", markersize = 3.3, linestyle = "None",
color = "orange", capsize = 3, label = "X-ray GRID 2gy")
p5, caps5, bars5 = plt.errorbar(meandose_5gy, np.mean(sf_5gy_dx05), yerr =
np.mean(cell_conf_5gy_dx05), marker = "^", markersize = 3.3, linestyle = "None",
color = "orange", capsize = 3, label = "X-ray GRID 5gy")
p6, caps6, bars6 = plt.errorbar(meandose_10gy,
np.mean(sf_10gy_dx05),yerr=np.mean(cell_conf_10gy_dx05), marker = "s", markersize =
3.3, linestyle = "None", color = "orange", capsize = 3, label = "X-ray GRID 10gy")
[bar.set_alpha(0.5) for bar in bars4];[cap.set_alpha(0.5) for cap in caps4]
[bar.set_alpha(0.5) for bar in bars5];[cap.set_alpha(0.5) for cap in caps5]
[bar.set_alpha(0.5) for bar in bars6];[cap.set_alpha(0.5) for cap in caps6]

#proton GRID:
p10, caps10, bars10 = plt.errorbar(0.6690,  0.9538848307795695 , yerr =
0.1337868945893167, marker = "o",markersize = 3.3, color = "darkslategray",
linestyle = "None", capsize = 3, label = "Proton GRID 2gy")
p11, caps11, bars11 = plt.errorbar(1.6726, 0.8300919575912646, yerr =
0.12317684617621223, marker = "^",markersize = 3.3, color = "darkslategray",
linestyle = "None", capsize = 3, label = "Proton GRID 5gy")
p12, caps12, bars12 = plt.errorbar(3.3451, 0.6977887550560387, yerr =
0.10832671286944257, marker = "s",markersize = 3.3, color = "darkslategray",
linestyle = "None", capsize = 3, label = "Proton GRID 10gy")
[bar.set_alpha(0.5) for bar in bars10];[cap.set_alpha(0.5) for cap in caps10]
[bar.set_alpha(0.5) for bar in bars11];[cap.set_alpha(0.5) for cap in caps11]
[bar.set_alpha(0.5) for bar in bars12];[cap.set_alpha(0.5) for cap in caps12]

plt.semilogy(np.linspace(0,10,1000), np.exp(log_sf(np.linspace(0,10,1000))), ":",
color ="gray", linewidth = 0.5, label = "LQ fit, X-ray open fields")
plt.fill_between(np.linspace(0,10,10000), np.exp(sf_upper_conf),
np.exp(sf_lower_conf), color= "0.9", label = "95% CI")

plt.ylabel("Mean survival fraction")
plt.xlabel("Mean dose received [Gy]")
plt.yscale("log")
plt.legend(ncol=2)
plt.show()



#predicted vs observed:
```

```
diff2gy_dx05 = 100. * np.abs(expected_2gy_dx05 - sf_2gy_dx05)/((expected_2gy_dx05 +
sf_2gy_dx05)/2.) #relative % diff.
diff5gy_dx05 = 100. * np.abs(expected_5gy_dx05 - sf_5gy_dx05)/((expected_5gy_dx05 +
sf_5gy_dx05)/2.)
diff10gy_dx05 = 100. * np.abs(expected_10gy_dx05 -
sf_10gy_dx05)/((expected_10gy_dx05 + sf_10gy_dx05)/2.)
diff2gy_dx15 = 100. * np.abs(expected_2gy_dx15 - sf_2gy_dx15)/((expected_2gy_dx15 +
sf_2gy_dx15)/2.)
diff5gy_dx15 = 100. * np.abs(expected_5gy_dx15 - sf_5gy_dx15)/((expected_5gy_dx15 +
sf_5gy_dx15)/2.)
diff10gy_dx15 = 100. * np.abs(expected_10gy_dx15 -
sf_10gy_dx15)/((expected_10gy_dx15 + sf_10gy_dx15)/2.)

plt.rcParams['legend.handlelength'] = 1.7
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1 = ax1.plot(x05[13:-10], expected_2gy_dx05[13:-10], "-k")
p2 = ax1.fill(np.NaN,np.NaN, "k", alpha=0.5)
ax1.fill_between(x05[13:-10], sf_conf_low_2gy_dx05[13:-10],
sf_conf_high_2gy_dx05[13:-10], color = "k",alpha = 0.5)
p3 = ax1.plot(x05[13:-10], sf_2gy_dx05[13:-10], color= "orange")
p4 = ax1.fill(np.NaN,np.NaN, "orange", alpha=0.5)
ax1.fill_between(x05[13:-10], sf_2gy_dx05[13:-10] - cell_conf_2gy_dx05[13:-10],
sf_2gy_dx05[13:-10] + cell_conf_2gy_dx05[13:-10], color = "orange", alpha = 0.5)
p0 = ax2.plot(x05[13:-10], diff2gy_dx05[13:-10], ":", color="gray")
plt.title("Predicted vs observed cell survival for 2 gy, counted with dx = 0.5 mm")
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)},loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((0.6,1.3))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1 = ax1.plot(x15[5:-4], expected_2gy_dx15[5:-4],"-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x15[5:-4], sf_conf_low_2gy_dx15[5:-4], sf_conf_high_2gy_dx15[5:-
4], color = "k",alpha = 0.5)
p3 = ax1.plot(x15[5:-4], sf_2gy_dx15[5:-4], color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 = ax2.plot(x15[5:-4], diff2gy_dx15[5:-4], ":", color="gray")
ax1.fill_between(x15[5:-4], sf_2gy_dx15[5:-4]- cell_conf_2gy_dx15[5:-4],
sf_2gy_dx15[5:-4] + cell_conf_2gy_dx15[5:-4], color = "orange", alpha = 0.5)
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"],  handler_map={tuple:
HandlerTuple(ndivide=None)},loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
```

```
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((0.6,1.3))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1 = ax1.plot(x05[13:-10], expected_5gy_dx05[13:-10], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x05[13:-10], sf_conf_low_5gy_dx05[13:-10],
sf_conf_high_5gy_dx05[13:-10] , color = "k", alpha =0.5)
p3 = ax1.plot(x05[13:-10], sf_5gy_dx05[13:-10],color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 = ax2.plot(x05[13:-10], diff5gy_dx05[13:-10], ":", color="gray")
ax1.fill_between(x05[13:-10], sf_5gy_dx05[13:-10] - cell_conf_5gy_dx05[13:-10],
sf_5gy_dx05[13:-10] + cell_conf_5gy_dx05[13:-10], color = "orange", alpha = 0.5)
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)}, loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((0.2,1.5))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p0 = ax2.plot(x15[5:-4], diff5gy_dx15[5:-4], ":", color="gray")
p1 =ax1.plot(x15[5:-4], expected_5gy_dx15[5:-4], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x15[5:-4], sf_conf_low_5gy_dx15[5:-4], sf_conf_high_5gy_dx15[5:-
4], color = "k", alpha = 0.5)
p3 = ax1.plot(x15[5:-4], sf_5gy_dx15[5:-4], color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 = ax2.plot(x15[5:-4], diff5gy_dx15[5:-4], ":", color="gray")
ax1.fill_between(x15[5:-4], sf_5gy_dx15[5:-4] - cell_conf_5gy_dx15[5:-4],
sf_5gy_dx15[5:-4] + cell_conf_5gy_dx15[5:-4], color = "orange", alpha = 0.5)
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)}, loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((0.2,1.5))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
```

```
p1 =ax1.plot(x05[13:-10], expected_10gy_dx05[13:-10], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x05[13:-10], sf_conf_low_10gy_dx05[13:-10] ,
sf_conf_high_10gy_dx05[13:-10] , color = "k", alpha = 0.5)
p3 = ax1.plot(x05[13:-10], sf_10gy_dx05[13:-10],color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 =ax2.plot(x05[13:-10], diff10gy_dx05[13:-10], ":", color="gray")
ax1.fill_between(x05[13:-10], sf_10gy_dx05[13:-10] - cell_conf_10gy_dx05[13:-10],
sf_10gy_dx05[13:-10] + cell_conf_10gy_dx05[13:-10], color = "orange", alpha = 0.5)
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)}, loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax2.set_ylim((0,190))
ax1.set_ylim((-0.1,1.4))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1= ax1.plot(x15[5:-4], expected_10gy_dx15[5:-4], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x15[5:-4], sf_conf_low_10gy_dx15[5:-4], sf_conf_high_10gy_dx15[5:-
4] , color = "k",alpha = 0.5)
p3 = ax1.plot(x15[5:-4], sf_10gy_dx15[5:-4], color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
ax1.fill_between(x15[5:-4], sf_10gy_dx15[5:-4] - cell_conf_10gy_dx15[5:-4],
sf_10gy_dx15[5:-4] + cell_conf_10gy_dx15[5:-4], color = "orange", alpha = 0.5)
p0 = ax2.plot(x15[5:-4], diff10gy_dx15[5:-4], ":", color="gray")
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)}, loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax2.set_ylim((0,190))
ax1.set_ylim((-0.1,1.4))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()
```

Predicted and observed cell survival protons

```
#finding and plotting the pedicted and observed cell survival for proton GRID:
```

```
import glob
import os
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.legend_handler import HandlerTuple
from scipy.signal import argrelextrema

def log_sf(dose):
    alpha, beta = [0.07441434687842306, 0.025813497191013064]
    return -alpha*dose - beta*dose**2

def interpolate_match(profile, conf, match):
    old_points = np.arange(len(profile))
    wanted_points = np.arange(len(match))
    interpol = np.interp(wanted_points, old_points, profile)
    interpol_conf = np.interp(wanted_points, old_points, conf)

    f1 = np.fft.fft(match); f0 = np.fft.fft(interpol)
    correlation = np.abs(np.fft.ifft((f0 * np.conj(f1))))
    shift = np.unravel_index(np.argmax(correlation), f0.shape)[0]

    shifted_profile = np.zeros(len(interpol) + shift)
    shifted_profile[:-shift] = interpol
    shifted_conf = np.zeros(len(interpol_conf) + shift)
    shifted_conf[:-shift] = interpol_conf
    return shifted_profile[shift:], shifted_conf[shift:]

sf1_2gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean2Gy-dx05.txt",
"r").readlines(), dtype = float)
sf1_5gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean5Gy-dx05.txt",
"r").readlines(), dtype = float)
sf1_10gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean10Gy-dx05.txt",
"r").readlines(), dtype = float)
cell_conf1_2gy_dx05 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std2Gy-
dx05.txt", "r").readlines(), dtype = float)/np.sqrt(6)
cell_conf1_5gy_dx05 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std5Gy-
dx05.txt", "r").readlines(), dtype = float)/np.sqrt(6)
cell_conf1_10gy_dx05 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std10Gy-
dx05.txt", "r").readlines(), dtype = float)/np.sqrt(6)

sf1_2gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean2Gy-dx15.txt",
"r").readlines(), dtype = float)
```

```
sf1_5gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean5Gy-dx15.txt",
"r").readlines(), dtype = float)
sf1_10gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/mean10Gy-dx15.txt",
"r").readlines(), dtype = float)
cell_conf1_2gy_dx15 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std2Gy-
dx15.txt", "r").readlines(), dtype = float)/np.sqrt(6)
cell_conf1_5gy_dx15 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std5Gy-
dx15.txt", "r").readlines(), dtype = float)/np.sqrt(6)
cell_conf1_10gy_dx15 =
1.96*np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/std10Gy-
dx15.txt", "r").readlines(), dtype = float)/np.sqrt(6)

#data for plotting with X-ray
print "Mean sf 2gy GRID", np.mean(sf1_2gy_dx05[15:90]), "conf",
np.mean(cell_conf1_2gy_dx05[15:90])
print "Mean sf 5gy GRID", np.mean(sf1_5gy_dx05[15:90]), "conf",
np.mean(cell_conf1_5gy_dx05[15:90])
print "Mean sf 10 gy GRID", np.mean(sf1_10gy_dx05[15:90]), "conf",
np.mean(cell_conf1_10gy_dx05[15:90])

x05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/xarray05.txt",
"r").readlines(), dtype = float)[:-3]
x15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/cell_arrays_proton/xarray15.txt",
"r").readlines(), dtype = float)[:-2]

dose5gy = np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose.txt",
"r").readlines(), dtype = float)
dose5gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx05.txt",
"r").readlines(), dtype = float)
dose5gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx15.txt",
"r").readlines(), dtype = float)
dose5gy_conf =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/conf_band.txt",
"r").readlines(), dtype = float)
dose5gy_conf_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx05.txt",
"r").readlines(), dtype = float)
dose5gy_conf_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx15.txt",
"r").readlines(), dtype = float)
```

```
dose2gy =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose2gy.txt",
"r").readlines(), dtype = float)
dose2gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx05_2gy.txt",
"r").readlines(), dtype = float)
dose2gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx15_2gy.txt",
"r").readlines(), dtype = float)
dose2gy_conf =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/conf_band2gy.txt",
"r").readlines(), dtype = float)
dose2gy_conf_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx05_2gy.txt",
"r").readlines(), dtype = float)
dose2gy_conf_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx15_2gy.txt",
"r").readlines(), dtype = float)

dose10gy =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose10gy.txt",
"r").readlines(), dtype = float)
dose10gy_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx05_10gy.txt",
"r").readlines(), dtype = float)
dose10gy_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/meandose_dx15_10gy.txt",
"r").readlines(), dtype = float)
dose10gy_conf =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/conf_band10gy.txt",
"r").readlines(), dtype = float)
dose10gy_conf_dx05 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx05_10gy.txt",
"r").readlines(), dtype = float)
dose10gy_conf_dx15 =
np.array(open("/Users/bjorgvh/Dropbox/Master/dose_proton/confband_dx15_10gy.txt",
"r").readlines(), dtype = float)

sf_upper_conf = np.array(open("/Users/bjorgvh/Dropbox/Master/upper_conf.txt",
"r").readlines(), dtype = float)
sf_lower_conf = np.array(open("/Users/bjorgvh/Dropbox/Master/lower_conf.txt",
"r").readlines(), dtype = float)
lqcurve = np.array(open("/Users/bjorgvh/Dropbox/Master/lqcurve.txt",
"r").readlines(), dtype = float)

expected_2gy_dx05 = np.exp(log_sf(dose2gy_dx05))
expected_5gy_dx05 = np.exp(log_sf(dose5gy_dx05))
expected_10gy_dx05 = np.exp(log_sf(dose10gy_dx05))

expected_2gy_dx15 = np.exp(log_sf(dose2gy_dx15))
expected_5gy_dx15 = np.exp(log_sf(dose5gy_dx15))
```

```
expected_10gy_dx15 = np.exp(log_sf(dose10gy_dx15))

sf_conf_low_2gy_dx05 = np.zeros(dose2gy_dx05.shape)
sf_conf_high_2gy_dx05 = np.zeros(dose2gy_dx05.shape)
for i in range(dose2gy_dx05.shape[0]):
    sf_conf_high_2gy_dx05[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose2gy_dx05[i]-dose2gy_conf_dx05[i]))).argmin()])
    sf_conf_low_2gy_dx05[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose2gy_dx05[i]+dose2gy_conf_dx05[i]))).argmin()])

sf_conf_low_5gy_dx05 = np.zeros(dose5gy_dx05.shape)
sf_conf_high_5gy_dx05 = np.zeros(dose5gy_dx05.shape)
for i in range(dose5gy_dx05.shape[0]):
    sf_conf_high_5gy_dx05[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose5gy_dx05[i]-dose5gy_conf_dx05[i]))).argmin()])
    sf_conf_low_5gy_dx05[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose5gy_dx05[i]+dose5gy_conf_dx05[i]))).argmin()])

sf_conf_low_10gy_dx05 = np.zeros(dose10gy_dx05.shape)
sf_conf_high_10gy_dx05 = np.zeros(dose10gy_dx05.shape)
for i in range(dose10gy_dx05.shape[0]):
    sf_conf_high_10gy_dx05[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose10gy_dx05[i]-dose10gy_conf_dx05[i]))).argmin()])
    sf_conf_low_10gy_dx05[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose10gy_dx05[i]+dose10gy_conf_dx05[i]))).argmin()])


sf_conf_low_2gy_dx15 = np.zeros(dose2gy_dx15.shape)
sf_conf_high_2gy_dx15 = np.zeros(dose2gy_dx15.shape)
for i in range(dose2gy_dx15.shape[0]):
    sf_conf_high_2gy_dx15[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose2gy_dx15[i]-dose2gy_conf_dx15[i]))).argmin()])
    sf_conf_low_2gy_dx15[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose2gy_dx15[i]+dose2gy_conf_dx15[i]))).argmin()])

sf_conf_low_5gy_dx15 = np.zeros(dose5gy_dx15.shape)
sf_conf_high_5gy_dx15 = np.zeros(dose5gy_dx15.shape)
for i in range(dose5gy_dx15.shape[0]):
    sf_conf_high_5gy_dx15[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose5gy_dx15[i]-dose5gy_conf_dx15[i]))).argmin()])
    sf_conf_low_5gy_dx15[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose5gy_dx15[i]+dose5gy_conf_dx15[i]))).argmin()])

sf_conf_low_10gy_dx15 = np.zeros(dose10gy_dx15.shape)
sf_conf_high_10gy_dx15 = np.zeros(dose10gy_dx15.shape)
for i in range(dose10gy_dx15.shape[0]):
    sf_conf_high_10gy_dx15[i] = np.exp(sf_upper_conf[(np.abs(lqcurve -
log_sf(dose10gy_dx15[i]-dose10gy_conf_dx15[i]))).argmin()])
    sf_conf_low_10gy_dx15[i]  = np.exp(sf_lower_conf[(np.abs(lqcurve -
log_sf(dose10gy_dx15[i]+dose10gy_conf_dx15[i]))).argmin()])
```

```
sf_2gy_dx05, cell_conf_2gy_dx05 = interpolate_match(sf1_2gy_dx05,
cell_conf1_2gy_dx05, expected_2gy_dx05)
sf_2gy_dx15, cell_conf_2gy_dx15 = interpolate_match(sf1_2gy_dx15,
cell_conf1_2gy_dx15, expected_2gy_dx15)
sf_5gy_dx05, cell_conf_5gy_dx05 = interpolate_match(sf1_5gy_dx05,
cell_conf1_5gy_dx05, expected_5gy_dx05)
sf_5gy_dx15, cell_conf_5gy_dx15 = interpolate_match(sf1_5gy_dx15,
cell_conf1_5gy_dx15, expected_5gy_dx15)
sf_10gy_dx05, cell_conf_10gy_dx05 = interpolate_match(sf1_10gy_dx05,
cell_conf1_10gy_dx05, expected_10gy_dx05)
sf_10gy_dx15, cell_conf_10gy_dx15 = interpolate_match(sf1_10gy_dx15,
cell_conf1_10gy_dx15, expected_10gy_dx15)

diff2gy_dx05 = 100. * np.abs(expected_2gy_dx05 - sf_2gy_dx05)/((expected_2gy_dx05 +
sf_2gy_dx05)/2.)
diff5gy_dx05 = 100. * np.abs(expected_5gy_dx05 - sf_5gy_dx05)/((expected_5gy_dx05 +
sf_5gy_dx05)/2.)
diff10gy_dx05 = 100. * np.abs(expected_10gy_dx05 -
sf_10gy_dx05)/((expected_10gy_dx05 + sf_10gy_dx05)/2.)
diff2gy_dx15 = 100. * np.abs(expected_2gy_dx15 - sf_2gy_dx15)/((expected_2gy_dx15 +
sf_2gy_dx15)/2.)
diff5gy_dx15 = 100. * np.abs(expected_5gy_dx15 - sf_5gy_dx15)/((expected_5gy_dx15 +
sf_5gy_dx15)/2.)
diff10gy_dx15 = 100. * np.abs(expected_10gy_dx15 -
sf_10gy_dx15)/((expected_10gy_dx15 + sf_10gy_dx15)/2.)

plt.rcParams['legend.handlelength'] = 1.7
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1 = ax1.plot(x05[10:-10], expected_2gy_dx05[10:-10], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x05[10:-10], sf_conf_low_2gy_dx05[10:-10],
sf_conf_high_2gy_dx05[10:-10], color = "k",alpha = 0.5)
p3 = ax1.plot(x05[15:-15], sf_2gy_dx05[15:-15], color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 = ax2.plot(x05[15:-15], diff2gy_dx05[15:-15], ":", color="gray")
ax1.fill_between(x05[15:-15], sf_2gy_dx05[15:-15] - cell_conf_2gy_dx05[15:-15],
sf_2gy_dx05[15:-15] + cell_conf_2gy_dx05[15:-15], color = "orange", alpha = 0.5)
plt.title("Predicted vs observed cell survival for 2 gy, counted with dx = 0.5 mm")
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)},loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((0.5,1.5))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
```

```
p1 = ax1.plot(x15[4:-4], expected_2gy_dx15[4:-4],"-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x15[4:-4], sf_conf_low_2gy_dx15[4:-4], sf_conf_high_2gy_dx15[4:-
4], color = "k",alpha = 0.5)
p3 = ax1.plot(x15[4:-5], sf_2gy_dx15[4:-5], color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 = ax2.plot(x15[4:-4], diff2gy_dx15[4:-4], ":", color="gray")
ax1.fill_between(x15[4:-5], sf_2gy_dx15[4:-5]- cell_conf_2gy_dx15[4:-5],
sf_2gy_dx15[4:-5] + cell_conf_2gy_dx15[4:-5], color = "orange", alpha = 0.5)
plt.title("Predicted vs observed cell survival for 2 gy, counted with dx = 1.5 mm")
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)},loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax2.set_ylim((0,190))
ax1.set_ylim((0.5,1.5))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1 = ax1.plot(x05[10:-10], expected_5gy_dx05[10:-10], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x05[10:-10], sf_conf_low_5gy_dx05[10:-10],
sf_conf_high_5gy_dx05[10:-10], color = "k", alpha =0.5)
p3 = ax1.plot(x05[15:-15], sf_5gy_dx05[15:-15],color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 = ax2.plot(x05[13:-13], diff5gy_dx05[13:-13], ":", color="gray")
ax1.fill_between(x05[15:-15], sf_5gy_dx05[15:-15]- cell_conf_5gy_dx05[15:-15],
sf_5gy_dx05[15:-15] + cell_conf_5gy_dx05[15:-15], color = "orange", alpha = 0.5)
plt.title("Predicted vs observed cell survival for 5 gy, counted with dx = 0.5 mm")
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)},loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((0.1,1.5))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1 =ax1.plot(x15[4:-4], expected_5gy_dx15[4:-4], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x15[4:-4], sf_conf_low_5gy_dx15[4:-4], sf_conf_high_5gy_dx15[4:-
4], color = "k", alpha = 0.5)
p3 = ax1.plot(x15[4:-5], sf_5gy_dx15[4:-5], color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 = ax2.plot(x15[4:-4], diff5gy_dx15[4:-4], ":", color="gray")
```

```
ax1.fill_between(x15[4:-5], sf_5gy_dx15[4:-5] - cell_conf_5gy_dx15[4:-5],
sf_5gy_dx15[4:-5] + cell_conf_5gy_dx15[4:-5], color = "orange", alpha = 0.5)
plt.title("Predicted vs observed cell survival for 5 gy, counted with dx = 1.5 mm")
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)},loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((0.1,1.5))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1 =ax1.plot(x05[10:-10], expected_10gy_dx05[10:-10], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x05[10:-10], sf_conf_low_10gy_dx05[10:-10] ,
sf_conf_high_10gy_dx05[10:-10] , color = "k", alpha = 0.5)
p3 = ax1.plot(x05[15:-15], sf_10gy_dx05[15:-15], color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 =ax2.plot(x05[13:-13], diff10gy_dx05[13:-13], ":", color="gray")
ax1.fill_between(x05[15:-15], sf_10gy_dx05[15:-15]- cell_conf_10gy_dx05[15:-15],
sf_10gy_dx05[15:-15]+ cell_conf_10gy_dx05[15:-15], color = "orange", alpha = 0.5)
plt.title("Predicted vs observed cell survival for 10 gy, counted with dx = 0.5
mm")
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)},loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((-0.1,1.5))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()

fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
p1= ax1.plot(x15[4:-4], expected_10gy_dx15[4:-4], "-k")
p2 = ax1.fill(np.NaN, np.NaN, "k", alpha=0.5)
ax1.fill_between(x15[4:-4], sf_conf_low_10gy_dx15[4:-4], sf_conf_high_10gy_dx15[4:-
4] , color = "k",alpha = 0.5)
p3 = ax1.plot(x15[4:-5], sf_10gy_dx15[4:-5], color= "orange")
p4 = ax1.fill(np.NaN, np.NaN, "orange", alpha=0.5)
p0 = ax2.plot(x15[4:-4], diff10gy_dx15[4:-4], ":", color="gray")
ax1.fill_between(x15[4:-5], sf_10gy_dx15[4:-5] - cell_conf_10gy_dx15[4:-5],
sf_10gy_dx15[4:-5] + cell_conf_10gy_dx15[4:-5], color = "orange", alpha = 0.5)
plt.title("Predicted vs observed cell survival for 10 gy, counted with dx = 1.5
mm")
```

```
plt.legend([p1[0],p3[0],(p2[0],p4[0]), p0[0]], ["Predicted survival", "Observed
survival", "95% confidence intervals", "Relative difference"], handler_map={tuple:
HandlerTuple(ndivide=None)},loc = "upper right")
ax1.set_xlabel("Position in flask [cm]")
ax1.set_ylabel("Survival fraction")
ax1.set_ylim((-0.1,1.5))
ax2.set_ylim((0,190))
ax2.set_ylabel("Relative Percentage Difference [%]")
plt.show()
```