THE SAHLGRENSKA ACADEMY
DEPARTMENT OF RADIATION PHYSICS

CHARACTERIZATION AND EVALUATION OF A NIPAM POLYMER GEL MRI DOSIMETER SYSTEM

M.Sc. thesis

Christian Waldenberg

Essay/Thesis:	30 hp
Program and/or course:	Medical Physics Program
Level:	Second Cycle
Semester/year:	At/2015
Supervisor:	Anna Karlsson Hauer[1,2], Sofie Ceberg[1], Christian Gustafsson[1]
Examiner:	Magnus Båth
Report no:	

[1] Department of Medical Radiation Physics, Skåne University Hospital, Lund
[2] Department of Medical Physics and Biomedical Engineering, Sahlgrenska University Hospital, Gothenburg

Abstract

Essay/Thesis:	30 hp
Program and/or course:	Medical Physics Program
Level:	Second Cycle
Semester/year:	At/2015
Supervisor:	Anna Karlsson Hauer, Sofie Ceberg, Christian Gustafsson
Examiner:	Magnus Båth
Report No:	
Keyword:	Gel dosimetry, 3D, NiPAM, MRI

Purpose: The normoxic polymer gel dosimeter based on N-isopropyl acrylamide (NiPAM) is a promising full 3D-dosimeter with high spatial resolution and near tissue equivalency. However, limited work have been done investigating this dosimeter. The dose response and reproducibility of a NiPAM dosimeter was investigated. The dose rate dependence and the effect of sequential irradiation was studied. Furthermore, the homogeneity of the R2 values across the MRI field of view was examined.

Theory: Polymer gel dosimeters are mostly composed of water and when exposed to ionizing radiation, radicals are formed in the process of radiolysis which in turn polymerize the gel via propagation reactions. The gels dose response is related to the degree of monomer polymerization. As the gel polymerize, the spin-lattice relaxation rate changes, making it possible to evaluate the 3D dose distribution within the gel with magnetic resonance imaging.

Method: The chemicals for the gel were mixed under controlled conditions and was left to set over night. The NiPAM gel samples were irradiated to different doses using a TrueBeam™ linear accelerator. The absorbed dose was evaluated using a FSE-based MRI sequence and statistical significance of the analyzed data was calculated. The analysis of the DICOM images was carried out using an in house developed software for image processing and data handling.

Result: The gel dosimeter was found to respond linearly to the absorbed dose. The reproducibility analysis of the NiPAM gel did not generate a conclusive result and need to be investigated further. The gel exhibited a dose rate dependence, as well as a dependence on the sequential irradiation scheme. A higher dose rate as well as a higher per sequence dose resulted in a lower dose response. Homogeneity analysis of the calculated R2 values across the MRI field of view demonstrated a maximum difference of 4.3% between calculated R2-values.

Contents

1 Introduction **1**

2 Materials and methods **4**
2.1 NiPAM gel preparation 5
2.2 Irradiation 5
2.3 MRI readout 6
2.4 Data processing 9

3 Results **11**

4 Discussion **14**

5 Conclusion **20**

Acknowledgments **20**

References **21**

Appendix **25**

A F-test analysis of linear regressions **25**

B Matlab code of in-house developed software **26**

1 Introduction

As radiation therapy is getting more complex with the use of modulated radiation beams, such as Intensity-Modulated Radiation Therapy (IMRT) and Volumetric-Modulated Arc Therapy (VMAT), in conjunction with respiratory gating and multi-leaf-collimator tracking, the need for well suited radiation dosimeters constantly increases. Many of the radiation beams today are dynamic, in the sense that the angle of the gantry, dose rate and the position of the multi-leaf-collimator are constantly changed while the radiation beam is active. There are international recommendations, e.g. [1, 2], to do pretreatment verification measurements for such treatments and the complex dose distribution calls for a dosimeter which can reflect the true three-dimensional dose distribution.

Commonly today, dosimeters used for dose distribution measurements contain diodes or ionization chambers positioned in an array in a plane or in orthogonal planes in a phantom volume. These dosimeters have a limitation in spatial resolution as the diodes or chambers integrates the ionization over an area or a volume and the full 3D dose is estimated from a limited number of measurement points [3, 4]. Furthermore, the diodes or chambers are placed millimeters or centimeters apart and the dose between the measurement points have to be reconstructed and calculated. This makes the uncertainties of 3D quality assurance, QA, depend on both accurate measurements and calculation algorithms provided by the vendor.

Gamma pass rate evaluation is commonly used to compare two dose distributions. The concept combines the percentage dose difference between measured and planned dose as well as the distance to agreement (DTA). DTA is the distance between measured data point and the nearest point in the calculated dose distribution that exhibits the same dose. The proportion of pixels that passes the gamma index test is then used as a basis for determining the QA result [5]. However, there is research that points out that gamma pass rate for virtual 3D dosimeters that utilize arrays of detectors arranged in common 3D geometries does not capture clinically relevant dosimetric differences [6]. One suggestion is to implement dose evaluation trough dose volume histogram (DVH) analysis. This requires a true 3D-dosimeter with high spatial resolution. Although the resolution of a polymer gel dosimeter depends on the readout technique, every sub-volume of the gel dosimeter can measure a unique dose, making the spatial resolution in polymer gel dosimeters unmatched in all three dimensions.

Already in 1950 the use of a gel dosimeter was suggested for radiation dosimetry. It was then shown that certain dyes changed color when exposed to ionizing radiation [7]. Thirty-four years later it was discovered that nuclear magnetic resonance relaxation properties of ferrous sulfate chemical dosimeter changed when exposed to radiation, which made it possible to evaluate the gel with magnetic resonance imaging, MRI, [8] However, due to diffusion of the ferric ions produced by irradiation of the Fricke-type gel dosimeters, the dose image obtained deteriorates [9]. In 1992 a new type of gel dosimeter was proposed by Maryanski et

al., which was based on the polymerization of acrylamide and N,N'-methylene-bis-acrylamide (bis) monomers infused in an aqueous agarose matrix [10]. The polymerization reaction occurred by cross-linking of the monomers induced by the free radical products of water radiolysis. The new gel was given the acronym BANANA and was a few years later refined by Maryanski et al. by replacing the gelling agent with gelatin [11]. This dosimeter was named PAG. Although the new polymer gel dosimeter did not have the diffusion problem of Fricke gels, thus having a stable post-irradiation dose distribution, the dosimeter had another major concern. Due to its nature, the polymer gel dosimeter was susceptible to atmospheric oxygen inhibition of the polymerization processes and therefore needed to be manufactured in an oxygen-free environment [12]. This limitation was overcome when an antioxidant, which bound atmospheric oxygen, was introduced in the recipe. The problem of oxygen inhibition during production was eliminated, allowing the product being manufactured in normal atmospheric environments. This type of gel dosimeters that were insusceptible to atmospheric oxygen became known as normoxic gels, in contrast to oxygen sensitive gels that are called hypoxic gels. The first normoxic polymer gel dosimeter was reported by Fong et al. [13] and became known as MAGIC. In the MAGIC dosimeter, the acrylamid was exchanged to methacrylic acid and as antioxidant the gel contained copper sulfate and ascorbic acid, commonly known as vitamin C. Subsequently other antioxidants were introduced in the manufacture of normoxic gels, including tetrakis (hydroxymethyl) phosphonium chloride (THPC) [14]. The PAG and MAGIC dosimeters which utilized THPC as an antioxidant were named nPAG and nMAG respectively.

Polymer gel dosimeters have high spatial accuracy, favorable dose precision and are near tissue-equivalent, i.e, the absorbing and scattering properties for a given radiation match those of a certain biological tissue [12, 15]. However despite the promising aspects of polymer gel dosimetry, it have yet to be a common dosimeter in clinical use [15]. There could be several reasons for this. A few of them discussed by Ibbott, G. S. [15] are imaging artifacts emerging during evaluation of the gel, linear energy transfer dependence and the fact that there has been a limited effort to create low and high-density gels to simulate different tissues e.g. lung tissue. Although it has been reported that the polymer gel dosimeter is not sensitive to temperature during irradiation [16], the temperature of the polymer gel during MRI-scanning has high impact on the data. [17]. Another major reason to why this type of dosimeter has low acceptance in clinics could be that it is time consuming to manufacture and analyse, and, additionally, the monomer toxicity of these dosimeters makes them hazardous and inconvenient to use and manufacture. As discussed by Senden et al. [16], there are other polymer gel formulations which are less toxic. In particular N-Isopropylacrylamide (NiPAM) gel dosimeters holds great promise [16, 18].

Polymer gel dosimeters are largely composed of water and when exposed to ionizing radiation, water molecules are mainly ionized to form $H_2O^+ + e^-$, and

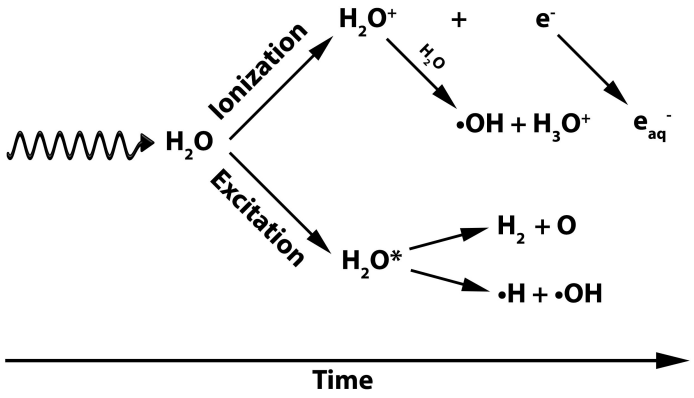

Figure 1: Schematic illustration of water radiolysis where radicals are marked with •.

are partially excited and broken up into •OH + •H or H_2 + O (Figure 1). The free electron ejected in the ionization process is solvated by a water molecule to form a hydrated electron, e_{aq} [19]. The radicals •OH, •H and e_{aq} react with the monomers in the gel, where products of one reaction supply reactants for the next reaction without outside intervention, forming polymer chains in a process called propagation. The chain growth comes to a halt only through the processes of termination and chain transfer. Termination occurs when two radicals meat, resulting in a destruction of the radicals, which prohibits further propagation. In the chain transfer process, a growing polymer chain abstracts and forms a covalent bond with a hydrogen atom from another molecule. This action expends the polymer radical and a new, and often more stable, radical is formed in the molecule in which the hydrogen atom was abstracted. Although possible, more stable radicals are not as prone as other radicals to further polymerization reactions [20].

Necessary ingredients to form a successful normoxic polymer gel, besides monomers and water, are a gelling agent and an antioxidant. Both are essential as the gelling agent, often gelatin, is required to maintain the spatial integrity of the gel and the antioxidant, usually THPC, binds the dissolved oxygen present in the gel. This prevents the oxygen from reacting with the the radicals as it is a reactive molecule which unbound will inhibit the monomer polymerization [12].

The degree of polymerization is related to the absorbed dose in the gel. As the polymerization takes place, the relaxation times of the protons alter, making it possible to evaluate the dose distribution with MRI. This is a technique which first was proposed by Maryanski et al. for polymer gel dose evaluation [21]. It was also reported that the spin-lattice relaxation rate (R1=1/T1) varied less over a certain dose range in the polymer gel dosimeter compared to the spin-spin

relaxation rate (R2=1/T2). This holds true for both 1.5 T and 3 T scans [22]. Hence, the most established MR sequences used for dose evaluation of polymer gel dosimeters probe R2 which yield a higher dose resolution. Fast Spin Echo (FSE) sequences, also known as Turbo Spin Echo (TSE), offer similar results for polymer gel dose evaluation compared to the more established Multi Spin Echo (MSE) sequences with greatly reduced acquisition times [23, 24].

Limited work have been done investigating the NiPAM dosimeter and the results of some studies contradict each other, e.g. the characteristics of the NiPAM gel dose rate dependence in the study by Farajollahi et al. [18] and Hsieh et al. [25]. The aim of this study was to investigate the dose response and reproducibility of a NiPAM gel dosimeter. The dose rate and sequential irradiation dependence was investigated. Furthermore, the homogeneity of calculated R2 across the MRI field of view (FOV) was examined. To evaluate the dose distribution in the gel dosimeter a 3T MRI scanner with an FSE based T2-mapping sequence was used. The analysis process was carried out using an in house developed software for image processing and data handling.

2 Materials and methods

Materials and supplies used for this study:

- Chemicals:
  - Deionized water
  - Gelatin, from porcine skin, (Sigma Aldrich, U.S.A.)
  - Antioxidant: Tetrakis hydroxymethyl phosphonium chloride, THPC, (80% solution in water, Sigma Aldrich, U.S.A.)
  - Monomer:
    - N-isopropyl acrylamide, NiPAM, (97% Sigma Aldrich, U.S.A.)
    - N,N'-methylene-bis-acrylamide, bis, ($\geq$ 98% Sigma Aldrich, U.S.A.)
- Laboratory glassware and equipment (heating mantle, temperature controller, magnetic stirrer)
- PVC phantom for irradiation of vials
- TrueBeam™ linear accelerator, Version 2.0, (Varian Medical Systems, Inc., 3100 Hansen Way, Palo Alto, U.S.A.)
- 3T General Electric MRI scanner, Model: Discovery MR750w 3.0T, (GE healthcare, Little Chalfont, United Kingdom). Accessories: Head coil, Model: 3.0T GEM HNU

- Matlab software, R2015b (8.6.0.267246), (Mathworks®, Massachusetts, U.S.A.)
- Eclipse™ Treatment Planning System, Version 10.0.28, Algorithm: AAA, (Varian Medical Systems, Inc., 3100 Hansen Way, Palo Alto, U.S.A.)

2.1 NiPAM gel preparation

The gel was prepared by heating deionized water together with gelatin (5% w/w) to 45 °C. When the gel had completely dissolved, first the NiPAM (3% w/w), and then the Bis (3% w/w) was added. Subsequently, the temperature was allowed to drop to 38 °C and the antioxidant (THPC) was added to a concentration of 5 mM. The gel was manufactured in a fume cupboard under normal atmospheric conditions and constantly mixed with a magnetic stirrer. The mixture was poured into 155 mm long glass vials equipped with plastic screw-tops holding 30 ml (Figure 2). The vials were left in dark to form a gel over night at room temperature, approximately 21 °C. Under the whole process, care was taken to minimize the gels exposure to ambient light as this could cause photopolymerization of the gel. The gel formula was inspired by Senden et al. [16]

2.2 Irradiation

While being irradiated, the vials containing gel were placed in a water filled phantom made out of a PVC-box of the size 27x26x26 cm^3 (h x w x d). The box was completely filled with tap water of room temperature and the vials were placed in an upright position in the water. The center of the vials was located at 7.7 cm depth from the outer edge of the PVC to provide adequate build-up for the 10 MV beam used (Figure 3). All vials were irradiated using a TrueBeam linear accelerator (Varian Medical Systems Inc, U.S.A.). A 10 MV beam with source-to-surface distance (SSD) 100 cm and a field size of 15x15 cm^2 was used to provide sufficient scattering conditions. The angle of the gantry was set to 90° and the collimator was set to 0°. The Multi Leaf collimator (MLC) was fully retracted. With the setup and settings used for irradiation of the gel, 100 MU delivered by the accelerator corresponds to 0.84 Gy absorbed in the gel. This was calculated using Eclipse™ Treatment Planning System and the AAA algorithm, (Varian Medical Systems, Inc.). The gel samples were irradiated approximately 24 hours after the manufacturing.

To characterize the dose rate dependence of the gel, 30 ml vials were irradiated to predetermined doses using different dose rates. A total of 33 vials were irradiated to 11 different absorbed doses spanning between 0.50 and 10 Gy (0.50, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0 and 10 [Gy]). Three vials were irradiated to each of the predetermined doses, each with a different dose rate, 100, 300 and 600 MU/min. There was a constant beam-on until the desired amount of monitor units had been delivered. One vial was left unirradiated and was used as a

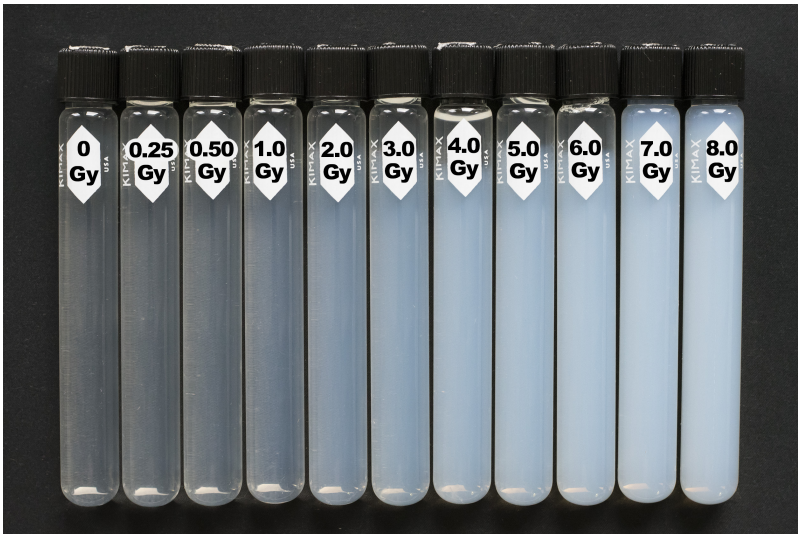

Figure 2: Vials containing NiPAM gel which have been irradiated to different absorbed doses using a dose rate of 600 MU/min.

measurement point for 0 Gy.

To characterize the sequential irradiation dependence of the NiPAM gel, a total number of 42 vials were irradiated. Doses for each beam sequence were 0.25, 0.50, 1.0, 2.0 and 4.0 Gy, see Figure 4 for the irradiation scheme. During irradiation, a dose rate of 600 MU/min was used. The beam-off time between two subsequent beams was 60 seconds. One vial was left unirradiated and was used as a measurement point for 0 Gy.

2.3 MRI readout

The gel was evaluated approximately 24 hours after irradiation using a Discovery 750w 3.0T MRI scanner (Discovery 750W General Electric Medical Systems, U.S.A.). There was a concern that heating could occur during scanning due to energy of the RF-pulses being absorbed by the gel [26]. To account for this, it would be advantageous to dummy-scan the vials for some time before the evaluation began and allow the gels to reach a temperature equilibrium. As time on the MRI was limited, the vials used for dose rate evaluation were pre-scanned for approximately 90 minutes and the vials used for sequential irradiation characterization were dummy-scanned for approximately 15 minutes.

The scanning sequence used for all scans, including the pre-scanning, was an FSE based T2-mapping sequence, named CartiGram, developed by GE [27]. The sequence acquires multiple echoes at different echo times, TE, and was originally intended to be used for scanning extra cellular cartilage. The gels were scanned with a total of 16 echoes with an inter echo time, ΔTE, of 80 ms (80 - 1280 ms)

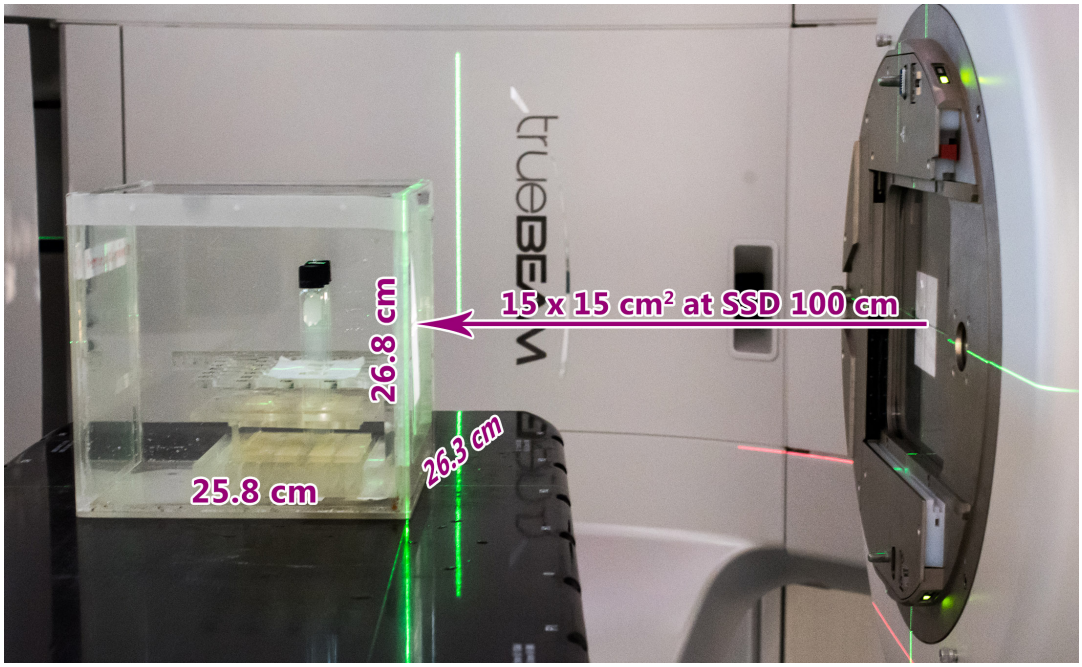

Figure 3: Experimental setup for irradiation of vials containing NiPAM gel. The PVC box was completely filled with tap water.

and a repetition time of 4000 ms. The ΔTE was changed by adjusting the GE CV protocol parameter ESP while the software was in researchmode.. The signal was measured while the vials were centered in a receiver head coil (3.0T GEM HNU).

The vials used for dose rate evaluation were scanned with a pixel size of 0.625 × 0.625 mm^2 and a slice thickness of 3 mm. The average of five acquisitions was obtained (number of excitations, NEX=5). To shorten the scan time, and still keep similar signal to noise ratio, the vials used for characterization of sequential irradiation dependence were scanned using a slice thickness of 5 mm and the average of two acquisitions (NEX=2) was obtained. Total scan time, excluding the dummy-scan, was 85.5 minutes and 34.2 minutes respectively.

The readout sequence produced series of 16 differently T2 weighted DICOM images, called base images. The images was prone to Gibbs artifacts, also known as truncation or ringing artifacts, due to the sharp discontinuities in the reconstructed images, as shown in Figure 5. The artifact occurs near high contrast boundaries and appears as ripples with alternating darker and brighter lines which diminish with distance from the boundary. This is a result of sampling a finite region of k-space but can be suppressed by multiplying the k-space data with a smoothing function or by increasing the size of the sampled k-space region by acquiring additional k-space data for higher spatial frequencies [28, 29]. For this reason, a matrix size of 256 x 256 was used for all acquisitions.

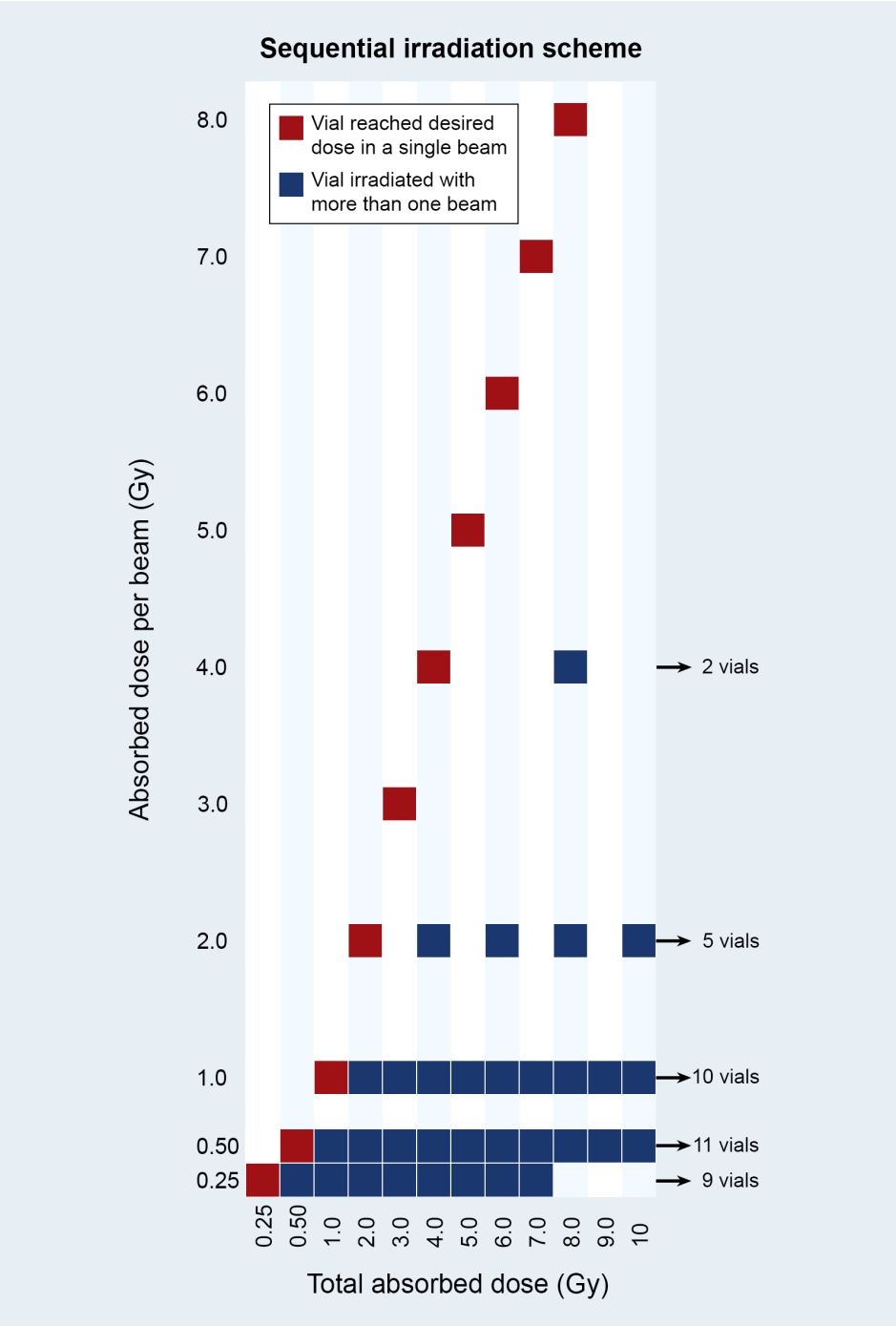

Figure 4: Sequential irradiation scheme used for NiPAM gel sequential irradiation characterization. Absorbed dose per beam versus total absorbed dose for 42 vials. The vials were irradiated with the dose rate of 600 MU/min and time between each subsequent beam was approximately 60 seconds.

The homogeneity of calculated R2 across the 16 x 16 cm^2 FOV was analyzed by scanning 36 vials filled with unirradiated NiPAM gel from the same batch.

2.4 Data processing

The images generated by the MRI were processed in an in-house developed software written in Matlab (Mathworks®, Massachusetts U.S.A.), (code for software provided in appendix).

The pixel values inside regions of interest (ROIs) were extracted from the DICOM images generated by the MRI system (Figure 5). The circular ROIs were placed on the vials in the image. They were made as large as possible (an area of approximately 350 pixels), so that sufficient data could be extracted. A gap of three to four pixels was left between the the ROI and the outer edge of the vial visible in the DICOM images. Individual pixels of the same position throughout the base images with different TE were used as data points. The R2 value was obtained by fitting the data points to an exponentially decaying curve, $S(TE) = S_0 \cdot e^{-R2 \cdot TE}$, where S is the pixel intensity, TE is the echo time and S_0 corresponds to the theoretical unrelaxed transversal magnetization at TE = 0 ms [30]. All regressions were calculated using the method of weighted nonlinear least squares using the levenberg-marquardt algorithm. The data points were weighted inversely proportional to the echo time, TE, as suggested by De Deene and Baldock [30]. The R2 value for each vial is defined as the mean calculated R2 from the pixels inside the corresponding ROI.

To evaluate the dose rate and sequential irradiation dependence, the calculated R2, and the corresponding standard deviation, were plotted as a function of the absorbed dose and linear regressions were fitted to the data points. All linear regressions were calculated using the method of Linear Least Square with the constraint that all regressions must pass trough their common data point (at zero Gy). This is a mutual datapoint as all regressions housed in the figure are using the same unirradiated vial as a source for data point for zero Gy.

An F-test of all linear regressions was performed to examine whether or not the regressions were statistically significantly different (Matlab code available in appendix). An F-test is most often used when comparing statistical models that have been fitted to a data set. The test can be used to test the significance of either a single or a series of regression coefficients and evaluates the ratio of two sample variances as evidence to test the null hypotesis that two population variances are equal [31, 32].

To calculate the uncertainties of the linear regressions, the method described by De Deene, et al. [33] was adopted. Standard error of the mean value of R2, SEM_{R2}, standard deviation of slope, σ_α, and standard deviation of intercept, σ_{R2_0}, were calculated by Equations 1, 2 and 3 respectively.

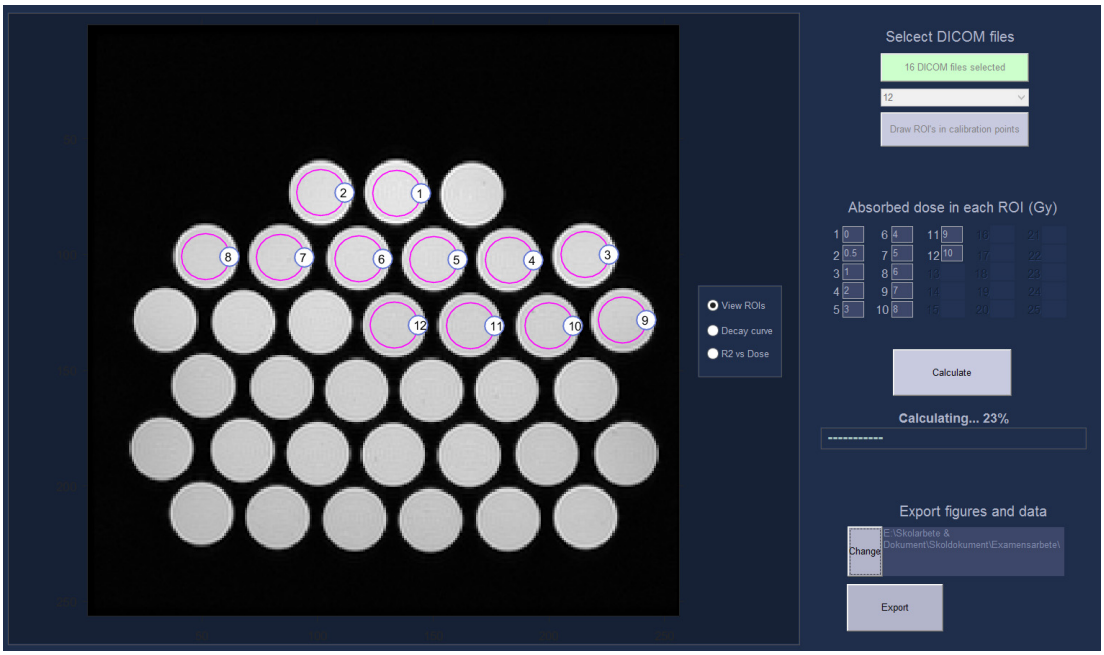

Figure 5: Screen shot of the in-house developed software for data analysis of the DICOM images from the MRI scanner. The image to the left shows ROIs on a transversal image of irradiated vials.

$$\mathrm{SEM}_{\mathrm{R2}} = \frac{\sigma_{\mathrm{cal}}}{\sqrt{\mathrm{N}_{\mathrm{ROI}}}} \tag{1}$$

$$\sigma_{\alpha} = \frac{\sigma_{\mathrm{cal}}}{\sqrt{\mathrm{N}_{\mathrm{ROI}}}} \sqrt{\frac{1}{\sum_{i=1}^{\mathrm{N}_{\mathrm{cal}}} (\mathrm{D}_i - \overline{\mathrm{D}})^2}} \tag{2}$$

$$\sigma_{\mathrm{R2}_0} = \frac{\sigma_{\mathrm{cal}}}{\sqrt{\mathrm{N}_{\mathrm{ROI}}}} \sqrt{\frac{1}{\mathrm{N}_{\mathrm{cal}}} + \frac{\overline{\mathrm{D}}^2}{\sum_{i=1}^{\mathrm{N}_{\mathrm{cal}}} (\mathrm{D}_i - \overline{\mathrm{D}})^2}} \tag{3}$$

Where

σ_{cal} is the standard deviation inside a ROI,
$\mathrm{N}_{\mathrm{ROI}}$ is the number of pixels inside a ROI,
$\mathrm{N}_{\mathrm{cal}}$ is the number of ROIs,
D_i is the absorbed dose inside ROI i,
$\overline{\mathrm{D}}$ is the mean dose of all ROIs.

Equations 2 and 3 are only valid if the standard error of the mean value of R2 (Equation 1) for each ROI are assumed to have the same value. If so, the slope

and intercept of the regression can be described to have a Gaussian distributed expectation value σ_α and σ_{R2_0} respectively [30]. During analysis, the maximum standard error of all ROIs was chosen as this ensures that a not to low level of uncertainty have been estimated.

3 Results

As expected, and reported elsewhere [34], the NiPAM gel turned opaque when irradiated (Figure 2).

A screen shot of the graphical user interface of the in-house developed software can be seen in Figure 5. The software allows the user to select a series of DICOM files to be used for analysis. The user is prompted to specify ROIs in the DICOM images and the absorbed dose for each ROI. With the information provided, R2 for each ROI is calculated. Furthermore, the software calculates and plots an exponentially decaying regression, with data points and their standard deviations, for a ROI (Figure 6). As the first three data points badly fit the regressions, they were omitted from the analysis (Figure 6). A second figure is displayed where R2, and its standard deviation, are plotted as a function of the absorbed dose. A progress bar is displayed during calculation. There is an option to export all figures as .png and the related data in a .tex file. This software is validated for DICOM files generated by 3T General Electric MRI scanner, Model: Discovery MR750w 3.0T, (GE healthcare, Little Chalfont, United Kingdom).

Linear regressions for three different dose rates (100, 200 and 600 MU$\cdot$ min^{-1}) were carried out. It was found that the dose response of the NiPAM gel was dependent on the dose rate (Figure 7). The slope of the 100 and 300 MU$\cdot$ min^{-1}-regression were 11.8% and 6.5% steeper respectively compared to the 600 MU$\cdot$ min^{-1}-regression. All three regressions were significantly different from each other ($p<0.05$).

It was found that the dose response of the gel dosimeter was dependent on sequential beam irradiation (Figure 8). That is when the sequential irradiation scheme in Figure 4 was used and time between each subsequent beam was approximately 60 seconds. A comparison of regressions and the statistical significance between the slopes of the regressions generated by pairwise F-test are listed in Table 1.

High R^2 values of the liner regressions indicates that the NiPAM gel responds linearly to the absorbed dose (Figure 8).

The NiPAM gel used for the dose rate dependence characterization and for sequential irradiation dependence characterization were mixed on two separate occasions. When the 600 MU$\cdot$ min^{-1}-regression from dose rate dependence analysis was compared to the regression of the single beam data points from the dose sequential irradiation dependence analysis, the slope and intercept of the regressions differed 2.7% and 1.1% respectively. The F-test p-value of the two

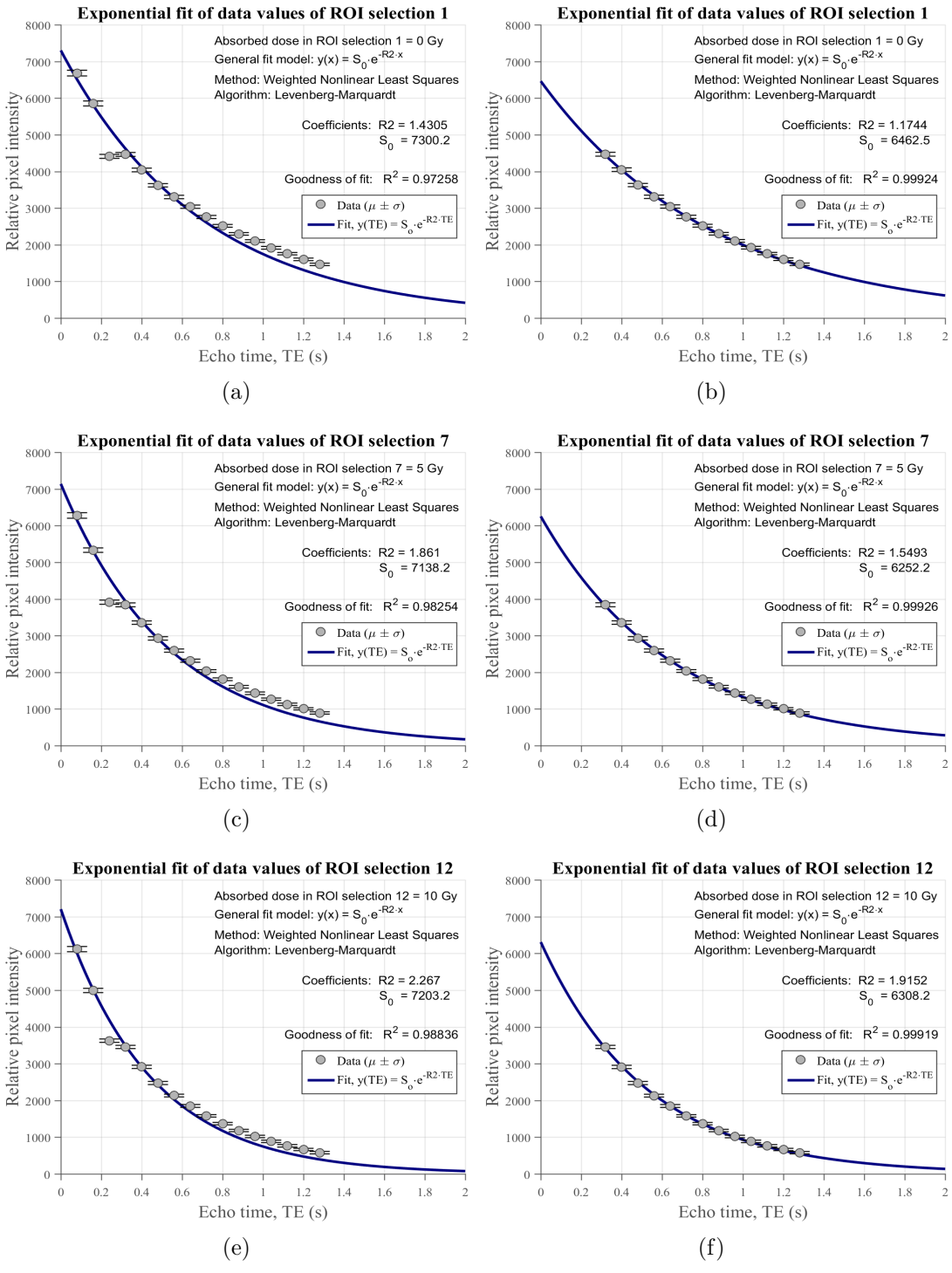

Figure 6: Data points and six regressions of three different ROIs of different doses. In (a), (c) and (e) all 16 available DICOM base images has been used to fit a decay curve. In (b), (d) and (f) the first three images (TE 80, 160 and 240 ms) has been omitted. The error bars represent two standard deviations for each measurement point.

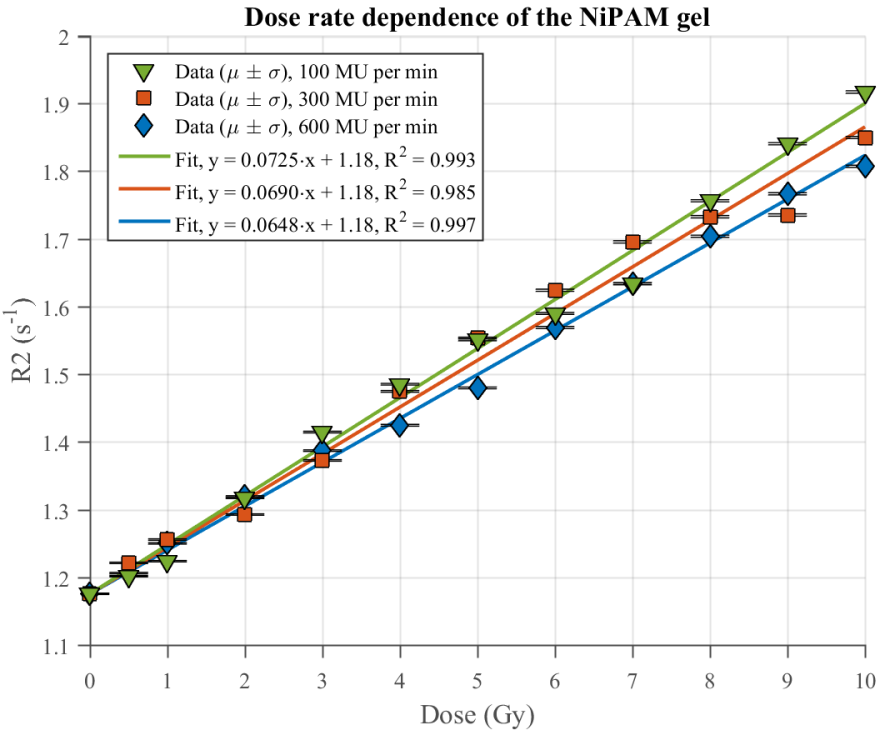

Figure 7: R2 versus absorbed dose for the dose rate dependence study of the NiPAM gel. Samples irradiated with dose rate 100, 300 and 600 MU $\cdot$ min^{-1}. Linear regressions were calculated using the method of Linear Least Squares.

regressions was 0.218, i.e. the regressions were not significantly different.

In the FOV-homogeneity analysis for the MRI scanner, the R2 values across the field of view differed maximum of 4.3% when 36 unirradiated vials filled with NiPAM gel from the same batch were scanned and analyzed (Figure 9). This might explain why several data points stray from their regressions. When investigating a few of them closer the 3 Gy data point of the 0.25 Gy per beam-regression (Figure 8) has the value 1.33 s^{-1} which is 3.9% below expected value from regression. During MRI, the sample was approximately located at the same position in the FOV as the vial in the bottom right corner in Figure 9. The calculated R2 value for this position was 2.1% below the R2 value of the vial chosen to represent 0.0% calculated error (Figure 9). Taking this into account would place the data point closer to its regression which after adjustment would deviate -1.8% from the regression. A similar analysis for the 6 Gy data point of the same 0.25 Gy per beam-regression was made. The data point had the value 1.63 s^{-1} which was 1.2% above its regression. The sample was located approximately at the same location in the FOV as the vial on the fifth row from the top and third position from the left in Figure 9. The calculated R2 value

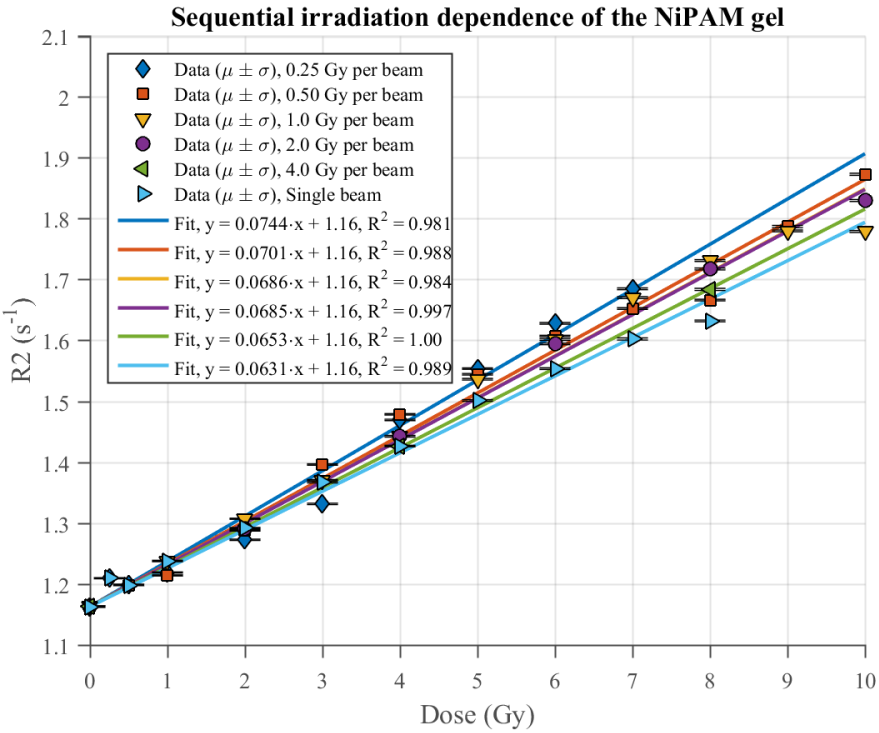

Figure 8: R2 versus absorbed dose for the sequential irradiation dependence study of the NiPAM gel. Samples irradiated with 600 MU · min^{-1}. Linear regressions were calculated using the method of Linear Least Squares.

for this area was 1.0% above the R2 value chosen to represent 0.0 % error in the MRI FOV. If adjusted, the data point would deviate +0.2% from its regression. The same procedure was made for all 100 MU per min data points and a new regression was made and compared to the original (Figure 10). The slope of the regression changed from 0.0725 to 0.0749 and the R^2 value was, after adjustment, 0.998, compared to $R^2 = 0.993$ before the adjustment.

4 Discussion

T2 weighted MRI DICOM images were generated and later analyzed in an in-house developed software. As the first three data points badly fit the regressions, they were excluded from the analysis (Figure 6). Although this phenomenon is clearly distinct in this study, it is hard to find in the literature. During FSE-sequences, refocusing pulses of less than 180° is often used. As a result the following signal will include contributions from stimulated echoes which will superimpose with the spin echoes. This may effect acquired data and cause artifacts [28]. Rewinding the transversal spin history signal will not remove stimulated

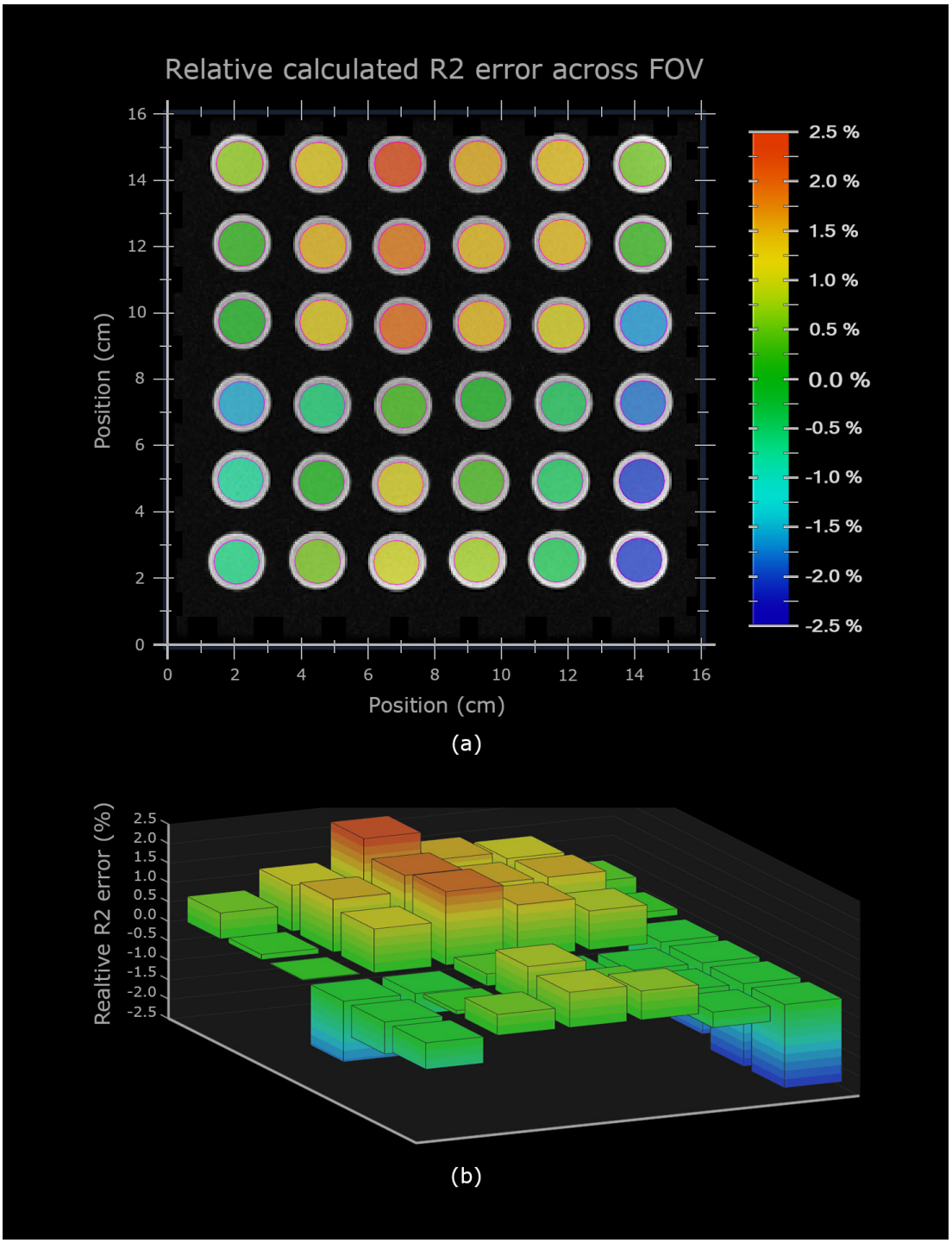

Figure 9: (a) shows a transversal image of 36 vials with color coded ROIs across the MRI field of view. The color indicates calculated R2 error relative to the normalized value where red is a high value and blue is a low value. (b) shows a 3D representation of the calculated error of the data retrieved from each ROI.

Table 1: Statistical significance and percentage difference between the slopes of regressions of the data which describes the sequential irradiation dependence of the NiPAM gel dosimeter. p-values are generated from pairwise F-test between slopes of regressions. p-value <0.05 indicates, with 95% confidence level, that the null hypothesis, that the slopes of the regressions are equal, can be rejected.

Beam (Gy/beam)	**Difference between the slopes of regressions (%)**	**Statistical significance (p-value)**
Single beam vs. 0.25	15.2	0.0002
Single beam vs. 0.50	10.0	0.002
Single beam vs. 1.0	8.0	0.014
Single beam vs. 2.0	7.9	0.005
Single beam vs. 4.0	3.4	0.351
4.0 vs. 0.25	12.2	0.021
4.0 vs. 0.50	7.0	0.119
4.0 vs. 1.0	4.8	0.309
4.0 vs. 2.0	4.7	0.067
2.0 vs. 0.25	7.9	0.032
2.0 vs. 0.50	2.4	0.399
2.0 vs. 1.0	0.2	0.955
1.0 vs. 0.25	7.8	0.046
1.0 vs. 0.50	2.1	0.434
0.50 vs. 0.25	5.7	0.070

echoes but will minimize artifacts. The CartiGram sequence may partly fail to suppress or rewind all stimulated echos and the possible imperfections of the sequence could appear as effects of deviation of the first data points.

The academic Matlab license offers a comprehensive selection of add-on toolboxes, several of which were used to develop the software for image processing and data handling in this study. The in house developed software was made compatible with the 3T GE MRI system used during this project. Some adjustment of the code may be necessary if a different MRI system is used. Matlab code is provided in Appendix.

As expected, the NiPAM gel turned opaque when irradiated. Although this was true for all batches of gel, the different batches of gel had different degree of transparency before being irradiated. This might have had an effect on the data used for evaluation of the reproducibility characteristics of the gel. Although the gels were mixed on two different occasions, the same recipe and irradiation process were used for both batches of gel. Statistical evaluation of regressions of the two data sets that are comparable (600 MU$\cdot$ min^{-1} data from dose rate dependence analysis and the single beam data from the sequential irradiation dependence analysis) produces the p-value 0.218 which indicates that the linear fits correlate, i.e., there is no statistical significant difference between the regressions. However,

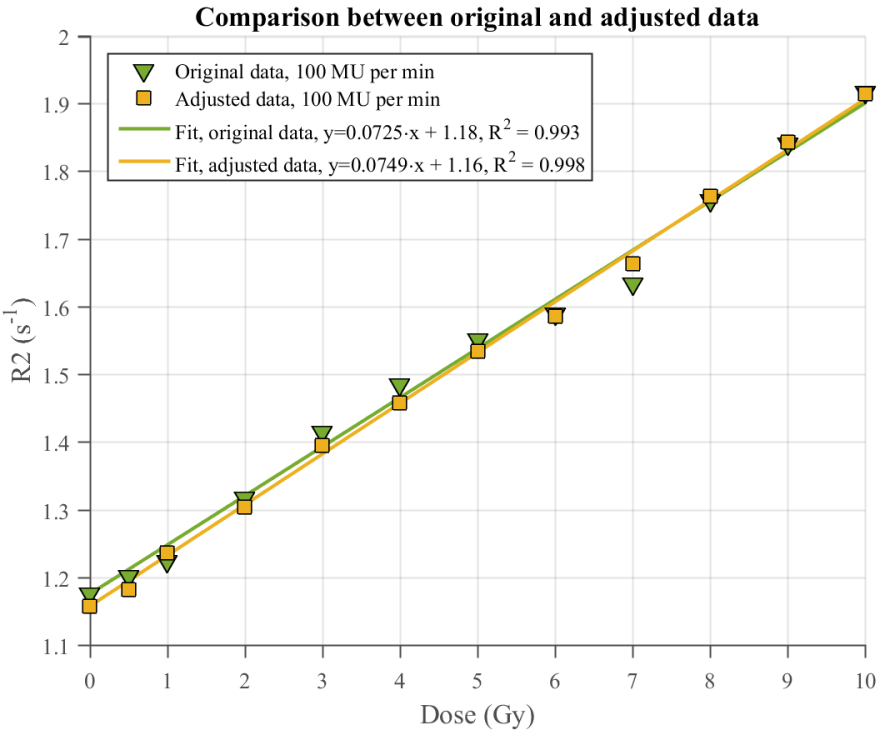

Figure 10: R2 versus absorbed dose for the comparison between original R2 data, and the same R2 data corrected for the estimated inhomogenous FOV of the MRI scanner. The regression is forced through the 0 Gy data point for both data sets (n.b. the 0 Gy data point is adjusted as well).

the p-value is too low to imply that the dose rate dependence characteristics between two separate batches of the gel are identical. This inconclusive result requires further investigation.

An F-test evaluation of the regressions comparing the dose rate dependence of the NiPAM gel (Figure 7) indicates that there is a difference between the regressions ($p<0.05$), i.e. the NiPAM gel is dependent on the irradiation dose rate. This is, of course, an unwanted characteristic as it limits the area of application of the gel. This result contradicts with the result in the study by Farajollahi et al. [18] but agrees with the study by Hsieh et al. [25].

It was found that the dose response of the gel is dependent on sequential beam irradiation (Figure 8). Table 1 lists p-values generated from pairwise F-test between slopes of regressions. The single, continuous beam, compared to the sequential beams were all statistically significantly uncorrelated with the exception of the single beam versus the 4 Gy per sequence-beam. Although the study by Karlsson et al [20] investigated different types of polymer gel dosimeters (nMAG and nPAG), their findings coincides with the result in this study.

Polymer gel dosimeters are mostly composed of water (≈89% w/w). When exposed to ionizing radiation, water molecules are either ionized or excited which produces free radicals in the process of water radiolysis. Many of the radicals react with monomers of the gel which add monomer units to the end of the polymer chain via propagation reactions. Chain growth comes to a halt when either two free radicals meet which result in the destruction of the free radicals so that further propagation cannot occur or when the radical undergo a chain transfer. The rate of production of free radicals, and thereby the concentration of radicals, is directly linked to the dose rate of the beam which the gel is being irradiated with. Unlike polymer growth termination reactions, the polymer chain growth process involves one polymer radical. As a result, the rate of reactions is proportional to the concentration of polymer radicals. However, termination reactions involve two radicals, consequently the incidence of termination reactions is proportional to the square of the radical concentration which makes termination reactions the dominant reaction when high concentration of radicals are present [20]. This might serve as an explanation to why the gel demonstrates dose rate dependence where a higher dose rate gives a lower dose response and vice versa. The sequential irradiation dependence of the NiPAM gel can be explained similarly; when the radiation beam is delivered in shorter bursts with no active beam in between, the radicals will approach high concentrations only for short periods and the concentration will drop in between beams. Compared to a single continuous beam, the average lifetime of the radicals generated by the sequential beam is longer, yielding more polymer formulation and increased dose response. It is likely that the sequential irradiation dependence of the gel would be different if time between the sequential beams had been altered. If time between the sequential beams had been reduced, the sequential irradiation dependence would probably be less pronounced.

A short analysis will emphasize the importance of dose rate and sequential irradiation dependence characterization. Assume that a radiation therapy treatment is given in 4 fields where 0.50 Gy is delivered to isocenter in each field using a dose rate of 600 MU/min. Due to sequential irradiation dependence of the gel, the R2 value retrieved would be R2 = 0.0701·(0.500·4) + 1.16 = 1.30 s^{-1} (numbers retrieved from the equation describing the 0.50 Gy per beam linear regression). If this value is inserted in to the single beam regression equation, the corresponding dose would be $x = \frac{R2-1.16}{0.0631} = 2.22$ Gy, which is 11% higher than the correct dose (Figure 11).

The calculated R2 values across the FOV of the MRI scanner differed with a maximum of 4.33%. Generally, the T2 values (T2 = 1/R2) were higher near the edges of the FOV (Figure 9). The 16 x 16 cm^2 FOV filled a substantial portion of the receiver head coil. Consequently, some vials were located close to the edge of receiver and the difference in distance between the receiving coil elements and the vials may have had an effect on the data generated by the MRI system. In future work, this should be investigated further. A possible coarse of action could be

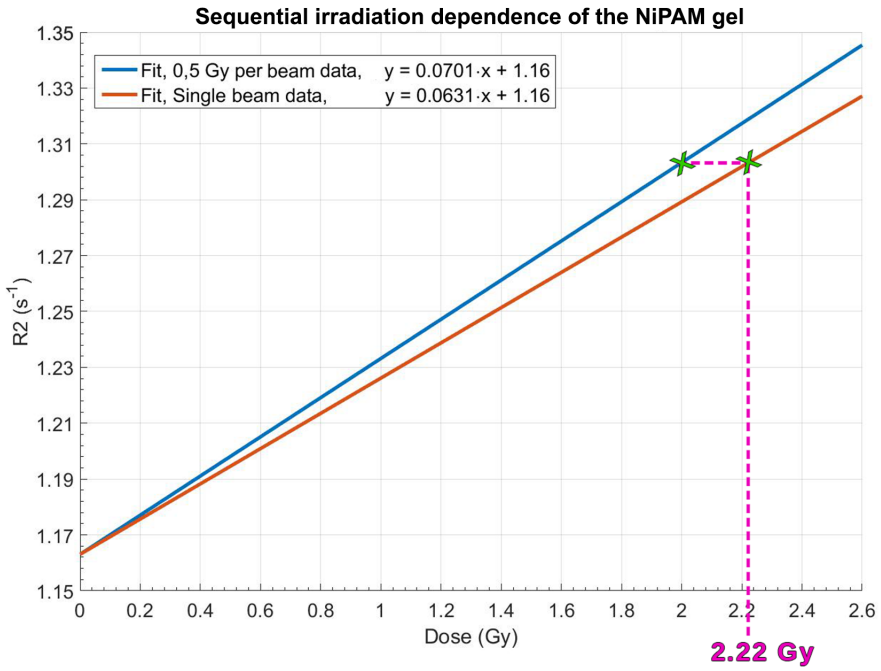

Figure 11: R2 versus absorbed dose for regressions to the 0.5 Gy per beam data and to the single beam data. The figure compares measured absorbed dose depending on wheter or not the sequential irradiation dependence of the NiPAM gel is taken to account. Sequential irradiation with 0.5 Gy per beam yields 11% higher measured dose in comparison to irradiation with a continues beam.

to investigate whether or not a different receiver coil, where the relative distance between the different vials and the edge of the receiver varied less, would produce similar results.

When the calculated R2 error across the MRI field of view is taken into account and the R2 vs dose -data points adjusted accordingly, the data points are more in line with their regressions. An adjustment was done for the data points of the 100 MU per min regression. The R^2 of the regression improved from 0.993 to 0.998 and the slope and the intercept of the regression altered (Figure 10). Although this result seems promising, the corrections were not done with high precision. The data points were corrected by visually comparing the location of the irradiated vial with a vial inside their FOV with a known error. Subsequently, the data point was adjusted accordingly to the calculated R2 error. Furthermore, depending on what R2 value is chosen to represent 0.0% error in the MRI FOV, the R2 value of the adjusted data points will be altered. This effectively changes the intercept of a regression if all associated data points are adjusted. However, this will not change the goodness of the fit as all data points

will be moved by the same amount .

As a result of the FOV inhomogeneity, it is likely that a large portion of the data points in this study deviates to a greater extent from its regression than they would if adjusted. As data for the different regressions in this study were retrieved from different parts of the FOV, it is likely that the calculated slopes of some regressions are either under- or overestimated. This might result in that some regressions, e.g. 1 and 2 Gy per beam - regressions (Figure 8), have practically no spread and are overlapping.

5 Conclusion

The dosimeter was found to respond linearly to the absorbed dose within the investigated range of doses and all regressions had a R^2-value greater then 0.98.

The NiPAM polymer gel dosimeter exhibited a dose rate dependence where a higher dose rate gave a lower dose response and vice versa. The gel was found to be dependent on the sequential irradiation scheme. A higher per beam dose resulted in a lower dose response.

The reproducibility test of the NiPAM gel did not generate a conclusive result and needs to be investigated further.

Homogeneity analysis of the MRI scanner field of view showed a maximum difference of 4.3% between calculated R2 values. This was shown to negatively affect the goodness of fit of calculated regressions as well as alter the regression coefficients.

The importance of dose rate and sequential irradiation dependence characterization has been emphasized. The measured dose by the NiPAM dosimeter for a radiation therapy treatment, consisting of four fields with a total delivered dose of 2 Gy, would be 11% higher if the sequential irradiation dependence would not have been taken into account.

Acknowledgments

I want to express my gratitude to my supervisors Anna Karlsson Hauer and Sofie Ceberg for the excellent guidance, feedback, and inspiring talks. Furthermore, I want to thank my supervisor Christian Gustafsson for the superb support in MRI, general feedback and advice. Finally, I want to thank Skåne University Hospital in Lund for the opportunity to use their facilities as well as all the medical physicist at Skåne University Hospital who have helped me practically in this study.

Thank you!

References

[1] Markus Alber, et al. (2008-First edition). Booklet 9: Guidelines for the verification of IMRT. Brussels (Belgium), ESTRO.

[2] Smilowitz, J. B., et al. (2015). The American Association of Physicists in Medicine, Medical Physics Practice Guideline 5.a.: Commissioning and QA of Treatment Planning Dose Calculations — Megavoltage Photon and Electron Beams.

[3] Low, D. (2015). "The importance of 3D dosimetry." Journal of Physics: Conference Series 573(1): 012009.

[4] Bäck, A. (2015). "Quasi 3D dosimetry (EPID, conventional 2D/3D detector matrices)." Journal of Physics: Conference Series 573(1): 012012.

[5] Low, D. A., et al. (1998). "A technique for the quantitative evaluation of dose distributions." Medical Physics 25(5): 656-661.

[6] Zhen, H., et al. (2011). "Moving from gamma passing rates to patient DVH-based QA metrics in pretreatment dose QA." Medical Physics 38(10): 5477-5489.

[7] Day M J and Stein G (1950). "Chemical effects of ionizing radiation in some gels." Nature 166 146 – 7

[8] Gore, J. C. and Y. S. Kang (1984). "Measurement of radiation dose distributions by nuclear magnetic resonance (NMR) imaging." Physics in Medicine and Biology 29(10): 1189.

[9] Olsson L E, Westrin B A, Fransson A and Nordell B (1992). "Diffusion of ferric ions in agarose dosimeter gels". Physics in Medicine and Biology 37 2243–52

[10] Maryanski M J, Gore J C and Schulz R J (1992). "3-D radiation dosimetry by MRI: solvent proton relaxation enhancement by radiation-controlled polymerisation and cross-linking in gels" Proceedings of the International Society for Magnetic Resonance in Medicine (New York)

[11] Maryanski, M. J., et al. (1994). "Magnetic resonance imaging of radiation dose distributions using a polymer-gel dosimeter." Physics in Medicine and Biology 39(9): 1437-1455.

[12] Baldock, C., et al. (2010). "Topical Review: Polymer gel dosimetry." Physics in Medicine and Biology 55(5): R1-R63.

[13] Fong, P. M., et al. (2001). "Polymer gels for magnetic resonance imaging of radiation dose distributions at normal room atmosphere." Physics in Medicine and Biology 46(12): 3105-3113.

[14] De Deene, Y., et al. (2002). "A basic study of some normoxic polymer gel dosimeters." Physics in Medicine and Biology 47(19): 3441-3463.

[15] Ibbott, G. S. (2004). "Applications of gel dosimetry." Journal of Physics: Conference Series 3(1): 58.

[16] Senden, R. J., et al. (2006). "Polymer gel dosimeters with reduced toxicity." Journal of Physics: Conference Series 56(1): 156.

[17] Pak, F., et al. (2013). "Influencing Factors on Reproducibility and Stability of MRI NIPAM Polymer Gel Dosimeter." BioImpacts 3(4): 163-168.

[18] Farajollahi, A. R., et al. (2014). "The basic radiation properties of the N-isopropylacrylamide based polymer gel dosimeter." International Journal of Radiation Research 12(4): 347-354.

[19] Yan, X. L. and R. Hino (2011). Nuclear Hydrogen Production Handbook, CRC Press.

[20] Karlsson, A., et al. (2007). "Dose integration characteristics in normoxic polymer gel dosimetry investigated using sequential beam irradiation." Physics in Medicine and Biology 52(15): 4697-4706.

[21] Maryanski, M. J., et al. (1993). "NMR relaxation enhancement in gels polymerized and cross-linked by ionizing radiation: a new approach to 3D dosimetry by MRI." Magnetic Resonance Imaging 11(2): 253-258.

[22] Berg, A., et al. (2001). "High-resolution polymer gel dosimetry by parameter selective MR-microimaging on a whole body scanner at 3T." Medical Physics 28(5): 833-843.

[23] Bankamp, A. and L. R. Schad (2003). "Comparison of TSE, TGSE, and CPMG measurement techniques for MR polymer gel dosimetry." Magnetic Resonance Imaging 21(8): 929-939.

[24] Baras, P., et al. (2005). "An evaluation of the TSE MR sequence for time efficient data acquisition in polymer gel dosimetry of applications involving high doses and steep dose gradients." Medical Physics 32(11):3339-3345.

[25] Hsieh, C.-M., et al. (2015). "The feasibility assessment of radiation dose of movement 3D NIPAM gel by magnetic resonance imaging." Radiation Physics and Chemistry 116: 142-146.

[26] De Deene, Y. and C. De Wagter (2001). "Artefacts in multi-echo T2 imaging for high-precision gel dosimetry: III. Effects of temperature drift during scanning." Physics in Medicine and Biology 46(10): 2697-2711.

[27] GE Healthcare. "CartiGram - Needle-Free Cartilage Assessment." Accessed 9 Dec, 2015, from `http://www3.gehealthcare.com/en/products/categories/magnetic_resonance_imaging/musculoskeletal_imaging/cartigram`.

[28] McRobbie, D. W., et al. (2006). MRI from Picture to Proton, Cambridge University Press.

[29] Patch, S, k-Space Data Preprocessing for Artifact Reduction in MR Imaging. Radiological Society of North America 2005 Scientific Assembly and Annual Meeting, November 27 - December 2, 2005 ,Chicago IL. Accessed 7 April, 2016, from `http://archive.rsna.org/2005/4405284.html`

[30] De Deene, Y. and C. Baldock (2002). "Optimization of multiple spin-echo sequences for 3D polymer gel dosimetry." Physics in Medicine and Biology 47(17): 3117-3141.

[31] LeBlanc, D. C. (2004). Statistics: Concepts and Applications for Science, Jones and Bartlett.

[32] Allen, M. P. (2007). Understanding Regression Analysis, Springer US.

[33] De Deene, Y., et al. (1998). "Mathematical analysis and experimental investigation of noise in quantitative magnetic resonance imaging applied in polymer gel dosimetry." Signal Processing 70(2): 85-101.

[34] Chang, Y.-J. and B.-T. Hsieh (2012). "Effect of Composition Interactions on the Dose Response of an N-Isopropylacrylamide Gel Dosimeter." PLoS ONE 7(10): e44905.

Appendix

A F-test analysis of linear regressions

The following Matlab code was used to examine whether or not one regression is statistically significantly different from another.

```
%% Example of calculation of statistical difference between regressions using F-test.
% In this example, it is calculated whether or not three regressions
% are statistically different using an F-test.

% Data to be fitted by a linear regression with a common intercept
Data_1 = [1.175567e+00 1.206170e+00 1.250241e+00 1.319476e+00 1.387112e+00 1.424674e+00 ...
    1.479821e+00 1.569052e+00 1.633737e+00 1.703962e+00 1.766617e+00 1.807266e+00]
Data_2 = [1.175567e+00 1.221358e+00 1.255802e+00 1.292779e+00 1.372666e+00 1.474769e+00 ...
    1.553464e+00 1.624159e+00 1.695528e+00 1.732764e+00 1.735245e+00 1.849735e+00]
Data_3 = [1.175800e+00 1.201568e+00 1.223862e+00 1.317342e+00 1.414560e+00 1.485280e+00 ...
    1.551189e+00 1.589880e+00 1.633637e+00 1.756303e+00 1.840559e+00 1.917006e+00]

y0=mean([Data_1(1) Data_2(1) Data_3(1)]);
Data_1 = Data_1-y0;
Data_2 = Data_2-y0;
Data_3 = Data_3-y0;

% x-data used to fit a linear regression
x_data=[0 5.000000e-01 1 2 3 4 5 6 7 8 9 10];

x_data=[x_data x_data x_data]';
y=[Data_1 Data_2 Data_3];
g = [ones(length(Data_1),1); 2*ones(length(Data_2),1); 3*ones(length(Data_3),1)];

% Get regression coefficient and other useful data to be used by the function linhyptest
% eye(3): Use no constant term in fitting function, i.e. linear regression forced through 0
s1 = regstats(y,[x_data, (g==2).*x_data, (g==3).*x_data],eye(3),{'beta' 'covb' 'fstat'});

% Get p-value that regressions of Data_2 and Data_3 are
%  statistically different from the regression of Data_1
[p] = linhyptest(s1.beta, s1.covb, [0;0], [0 1 0;0 0 1], s1.fstat.dfe);

% Get regression coefficient and other useful data to be used by the function linhyptest
% eye(3): Use no constant term in fitting function, i.e. linear regression forced through 0
[b12] = regstats(y,[x_data, (g==1).*x_data, (g==2).*x_data],eye(3),{'beta' 'covb' 'fstat'});
[b13] = regstats(y,[x_data, (g==1).*x_data, (g==3).*x_data],eye(3),{'beta' 'covb' 'fstat'});

%% Pairwise test (p-values)

% Null hypothesis:
% The difference between slope 2 and 3 is equal to zero, i.e. the slopes are identical
[p23] = linhyptest(b12.beta, b12.covb, 0, [0 0 1], b12.fstat.dfe)

% Null hypothesis:
% The difference between slope 1 and 3 is equal to zero, i.e. the slopes are identical
[p13] = linhyptest(b12.beta, b12.covb, 0, [0 1 0], b12.fstat.dfe)

% Null hypothesis:
% The difference between slope 1 and 2 is equal to zero, i.e. the slopes are identical
[p12] = linhyptest(b13.beta, b13.covb, 0, [0 1 0], b13.fstat.dfe)
```

B Matlab code of in-house developed software

The following code is the complete code required to run the software except the code required for layout of the graphical user interface. If the reader is interested, a copy of the Matlab .fig file can be provided upon request. All code is written in Matlab R2015b (8.6.0.267246) and is compatible with, but not limited to, that version.

```
function varargout = R2_calculate(varargin)
% R2_CALCULATE MATLAB code for R2_calculate.fig

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @R2_calculate_OpeningFcn, ...
                   'gui_OutputFcn',  @R2_calculate_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before R2_calculate is made visible.
function R2_calculate_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for R2_calculate
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = R2_calculate_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;
global flag_CorrectFiles flag_CorrectCalibrationPoints ...
    flag_break flag_DoneROI flag_NotCalculate

flag_CorrectFiles=0;
flag_CorrectCalibrationPoints=0;
flag_break=1;
flag_DoneROI=0;
flag_NotCalculate =1;


% --- Executes on button press in pushbutton_files.
function pushbutton_files_Callback(hObject, eventdata, handles)
```

```
global flag_CorrectFiles flag_CorrectCalibrationPoints fileNames dir flag_break

flag_CorrectFiles=0;
[fileNames,dir]=uigetfile('*.dcm','MultiSelect','on');
set(handles.text_exportDir,'string',dir)

a=size(fileNames);
if a(1) == a(2)
    flag_break=1;
    set(handles.pushbutton_files,'String','Select DICOM files')
    set(handles.pushbutton_files,'BackgroundColor','[.702 .710 .796]')
    set(handles.pushbutton_drawROI,'Enable','off')
    return

elseif iscell(fileNames) == 0
    flag_break=1;
    set(handles.pushbutton_files,'String','Select at least two DICOM files')
    set(handles.pushbutton_files,'BackgroundColor','[1 .7 .7]')

    for i = 1:4
        for i = 1:20
            i=((i+70)/90)-.1;
            i=[1 i i];
            set(handles.pushbutton_files,'BackgroundColor',i)
            pause(.01)
        end
        for i = 20:-1:1
            i=((i+70)/90)-.1;
            i=[1 i i];
            set(handles.pushbutton_files,'BackgroundColor',i)
            pause(.01)
        end
    end

    return
end


flag_CorrectFiles=1;
set(handles.pushbutton_files,'String',[num2str(length(fileNames)) ' DICOM files selected'])
set(handles.pushbutton_files,'BackgroundColor','[.8 1 .8]')


if flag_CorrectFiles==1 && flag_CorrectCalibrationPoints ==1
    flag_break=0;
    set(handles.pushbutton_drawROI,'Enable','on')
    while flag_break == 0
        for i = 20:-1:1
            i=(i/150);
            i=[.702-.05+i .710-.05+i .796-.05+i];
            set(handles.pushbutton_drawROI,'BackgroundColor',i)
            pause(.035)
        end
        for i = 1:20
            i=(i/150);
            i=[.702-.05+i .710-.05+i .796-.05+i];
            set(handles.pushbutton_drawROI,'BackgroundColor',i)
            pause(.030)
        end
    end
else
    set(handles.pushbutton_drawROI,'Enable','off')
end
```

```
% --- Executes on button press in pushbutton_drawROI.
function pushbutton_drawROI_Callback(hObject, eventdata, handles)

global fileNames dir NumberOfROIs pixelValueROI EchoTime flag_break ...
    Dose flag_DoneROI flag_NotCalculate flag_breakCalculate figROI Image

flag_break=1;
flag_breakCalculate = 1;
set(handles.pushbutton_drawROI,'Enable','off')
set(handles.CalibrationPoints,'Enable','Off')
set(handles.pushbutton_files,'Enable','off')
pause(.3)

% Clear some variables
 pixelValueROI={};
 EchoTime=0;


for i = 1:length(fileNames)
    % Saves multiple images in cell.
    I{i}=dicomread(fullfile(dir,fileNames{i}));

    % Saves DICOM info from all selected DICOM files.
    DI{i}=dicominfo(fullfile(dir,fileNames{i}));

    % Saves Echo Time from each selected DICOM file.
    EchoTime(i)=DI{1,i}.EchoTime;
end

% Code required to view DICOM image with higher contrast for easier ROI
% placement. This will not effect calculated values.
a=I{1,1};
a=double(a);
a=a/max(max(a))*255;
a=uint8(a);
axes(handles.axes1)
figROI(1)=imshow(a);

% Lets you specify ROIs and saves pixel values of selected ROIs in an
% array. The array contains n matrices where n is the number of specified
% ROIs. Every row in each matrix represents values of a pixel throughout
% the DICOM series (base images) which changes with TE.

for j = 1:NumberOfROIs

   if j==1
       %Show text in GUI
        set(handles.text_DragAndDrop,'Visible','On')

        ROI = imellipse;

        %Hide text in GUI
        set(handles.text_DragAndDrop,'Visible','Off')

   else
        ROI = imellipse(gca,[length(a)/2 length(a)/2 max(posROI)-min(posROI)]);
   end

    % Blocks execution of the MATLAB command line until you finish
    % positioning the ROI object. You indicate completion by
    % double-clicking on the ROI object. Also saves the position and
    % shape of ROI.
    posROI = wait(ROI);

    % Returns a mask that is the same size as the input image with 1s
```

```
    % inside the ROI object and 0s everywhere else.
    mask = createMask(ROI);

    delete(ROI)
    hold on
     plot(posROI(:,1),posROI(:,2),'m','LineWidth',1);
     plot(posROI(1,1),posROI(1,2),'o','Color','[.39 .47 .84]', ...
         'MarkerSize',16,'LineWidth',1,'markerfa','w');
     text(posROI(1,1)-2,posROI(1,2)-.2,num2str(j));
     Temp_pixelValueROI=[];

        for i = 1:length(fileNames)
            size(nonzeros(double(mask).*double(I{1,i})));
            Temp_pixelValueROI(:,i)= ...
                (nonzeros(double(mask).*double(I{1,i}+1))) -1;

        end

        % Using an array to save pixel values of ROI because ROI seizes may
        % not be equal. Saves pixel values of each ROI in its own cell.
        pixelValueROI{j}=Temp_pixelValueROI;
end
%Image = getimage(handles.axes1);
F = getframe(handles.axes1);
Image = frame2im(F);
flag_DoneROI=1;
%flag_NotCalculate=1;
set(handles.text_SpecifyDose,'Visible','On')
set(handles.text_OK,'Visible','On')

if flag_NotCalculate == 0 && length(Dose)== NumberOfROIs && flag_DoneROI==1
    flag_breakCalculate=0;
    set(handles.pushbutton_Calculate,'Enable','On')
    set(handles.text_SpecifyDose,'Visible','Off')
end

while flag_breakCalculate == 0
    for i = 20:-1:1
        i=((i+180)/200)-.07;
        i=[i i i];
        set(handles.pushbutton_Calculate,'BackgroundColor',i)
        pause(.035)
    end
    for i = 1:20
        i=((i+180)/200)-.07;
        i=[i i i];
        set(handles.pushbutton_Calculate,'BackgroundColor',i)
        pause(.045)
    end
end



% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)

set(allchild(handles.axes1),'visible','on');
```

```
set(handles.axes1,'visible','on');

set(allchild(handles.axes2),'visible','off');
set(handles.axes2,'visible','off');

set(allchild(handles.axes3),'visible','off');
set(handles.axes3,'visible','off');



% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)

set(allchild(handles.axes1),'visible','off');
set(handles.axes1,'visible','off');

set(allchild(handles.axes2),'visible','on');
set(handles.axes2,'visible','on');

set(allchild(handles.axes3),'visible','off');
set(handles.axes3,'visible','off');


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)

set(allchild(handles.axes1),'visible','off');
set(handles.axes1,'visible','off');

set(allchild(handles.axes2),'visible','off');
set(handles.axes2,'visible','off');

set(allchild(handles.axes3),'visible','on');
set(handles.axes3,'visible','on');



% --- Executes on selection change in CalibrationPoints.
function CalibrationPoints_Callback(hObject, eventdata, handles)

global flag_CorrectFiles flag_CorrectCalibrationPoints NumberOfROIs flag_break Dose
flag_CorrectCalibrationPoints = 0;
NumberOfROIs=(get(hObject,'Value'));

set(handles.CalibrationPoints,'backgroundcolor',[.702 .710 .796])
handlesStructure = guihandles(gcf);


for i= 1:NumberOfROIs
    h = handlesStructure.(sprintf('edit%d', i));
    Dose(i)= str2double(get(h, 'String'));
end

Dose=Dose(1:NumberOfROIs);

a=isnan(Dose);
for i=1:length(Dose)
   if a(i)==1 && length(Dose)>=i
       Dose=Dose(1:(i-1));

   elseif length(Dose) <2
       Dose=[];
   end
```

```
end

for i= 25:-1:1
    h = handlesStructure.(sprintf('edit%d', i));
            set(h, 'Enable','on')
    h = handlesStructure.(sprintf('text%d', i));
            set(h, 'Enable','on')
    h = handlesStructure.(sprintf('cover%d', i));
            set(h, 'visible','off')
end

if NumberOfROIs == 1
   flag_break=1;
   set(handles.edit1, 'Enable','off')
   set(handles.edit1,'String','')
   set(handles.text1, 'Enable','off')
   set(handles.cover1,'visible','on')
end

for i = 25:-1:1
    if NumberOfROIs < i
            h = handlesStructure.(sprintf('edit%d', i));
            set(h, 'Enable','off')
            set(h,'String','')
            h = handlesStructure.(sprintf('text%d', i));
            set(h, 'Enable','off')
                h = handlesStructure.(sprintf('cover%d', i));
            set(h, 'visible','on')

    end
end

if NumberOfROIs > 1
    flag_CorrectCalibrationPoints = 1;

    if flag_CorrectFiles==1 && flag_CorrectCalibrationPoints ==1
        set(handles.pushbutton_drawROI,'Enable','on')
        flag_break=0;
    end

    while flag_break == 0
        for i = 20:-1:1
            i=(i/150);
            i=[.702-.05+i .710-.05+i .796-.05+i];
            set(handles.pushbutton_drawROI,'BackgroundColor',i)
            pause(.035)
        end
        for i = 1:20
            i=(i/150);
            i=[.702-.05+i .710-.05+i .796-.05+i];
            set(handles.pushbutton_drawROI,'BackgroundColor',i)
            pause(.030)
        end
    end

else
    set(handles.pushbutton_drawROI,'Enable','off')
end



% --- Executes on button press in pushbutton_Calculate.
function pushbutton_Calculate_Callback(hObject, eventdata, handles)

global EchoTime NumberOfROIs pixelValueROI flag_breakCalculate ...
```

```
    Dose gof_plot2 gof_plot3 std_error_mean_R2 x co2 co3 weight ...
    mean_R2_ROI std_slope std_intercept

flag_breakCalculate =1;
set(handles.text_OK,'Visible','Off')
set(handles.pushbutton_drawROI,'Enable','off')

% Vector required to make a weighted least square fit.
weight = 1./EchoTime;

% Weighted least square fit using Levenberg-Marquardt algorithm. Fits data
% points from each pixel in each ROI to s*exp(-r*x) and saves values of
%"r" for each fit.
for k= 1:NumberOfROIs
    Temp_R2=zeros(1,length(pixelValueROI{k})); %Prellocate memory
    for m=1:length(pixelValueROI{k})

    % Prepare data to fit.
    [xData, yData, weights] = ...
        prepareCurveData( EchoTime/1000, pixelValueROI{k}(m,:), weight );

    % Set up fittype and options.
    ft = fittype( 's*exp(-r*x)', 'independent', 'x', 'dependent', 'y' );
    opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
    opts.Algorithm = 'Levenberg-Marquardt';
    opts.Display = 'Off';
    opts.StartPoint = [0.0025*1000 3000];
    opts.Weights = weights;

    % Fit model to data.
    [fitresult, gof] = fit( xData, yData, ft, opts );

    % Save value of "r" in R2
    FittValues=coeffvalues(fitresult);
    Temp_R2(m)=FittValues(1);

    % Code for displaying and updating a waitbar.
    perc = round(100/NumberOfROIs*(k-1) + ...
           (100/NumberOfROIs)/length(pixelValueROI{k})*m);
    loading(hObject, eventdata, handles, perc)

    end

    R2{k}=Temp_R2;

end

%-------------------------------------------------------------------------%
%  PLOTs  %%%%  PLOTs  %%%% PLOTs  %%%%  PLOTs  %%%% PLOTs  %%%% PLOTs  %
%-------------------------------------------------------------------------%

set(handles.uibuttongroup,'visible','on')


% Code required for plot off a exponential curve with datapoints.
    % Prepare data to fit.
    [xData, yData, weights] = ...
        prepareCurveData( EchoTime/1000, mean(pixelValueROI{1}), weight );

    % Set up fittype and options.
    ft = fittype( 's*exp(-r*x)', 'independent', 'x', 'dependent', 'y' );
    opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
    opts.Algorithm = 'Levenberg-Marquardt';
    opts.Display = 'Off';
    opts.StartPoint = [0.0025*1000 3000];
```

```
    opts.Weights = weights;

    % Fit model to data.
    [fitresult_plot2, gof_plot2] = fit( xData, yData, ft, opts );

    % Save data.
    co2=coeffvalues(fitresult_plot2);


    axes(handles.axes2)
    set(handles.axes2,'color','white')
    plot_decay(hObject, eventdata, handles, co2, gof_plot2, EchoTime, pixelValueROI, Dose);
    set(gca                         , ...
     'XColor'      , [.7 .7 .7] , ...
     'YColor'      , [.7 .7 .7] );
    hTitle  = title ('Exponential fit of data values of ROI selection 1');
    set( hTitle, 'Color', [.7 .7 .7]);



% Code required for plot off Dose vs mean_R2 with error bars and linear
% regression.

    % Calculate mean and standard deviation of R2 for pixels inside selected
    % ROIs. Also Calculate the standard error of mean of R2.
    mean_R2_ROI=[];
    for n=1:length(R2)
        mean_R2_ROI(n)= mean(R2{n});
        std_R2_ROI(n)= std(R2{n});
        std_error_mean_R2(n)=std_R2_ROI(n)./ sqrt(length(pixelValueROI{n}));
    end

    % Calculate standard deviation of slope (X) for linear regression (of the
    % form Y= ax + b) of R2 vs dose. Worst case scenario of std when using
    % "max(std_error_mean_R2)"
    std_slope = max(std_error_mean_R2)/sqrt(sum((Dose-mean(Dose)).^2));

    %Calculating standard deviation of intercept
    std_intercept=max(std_error_mean_R2)*sqrt( (1/length(Dose)) + ...
                 ((mean(Dose)^2)/sum(   ( Dose-mean(Dose) ).^2)) );

    % Fit
    [xData, yData] = prepareCurveData( Dose, mean_R2_ROI );

    % Set up fittype and options.
    ft = fittype( 'poly1' );
    opts = fitoptions( 'Method', 'LinearLeastSquares' );
    %opts.Robust = 'LAR';

    % Fit model to data.
    [fitresult_plot3, gof_plot3] = fit( xData, yData, ft, opts );

    % Save fit data
    co3=coeffvalues(fitresult_plot3);

    % Make the plot
    axes(handles.axes3)
    set(handles.axes3,'color','white')
    plot_linear(hObject, eventdata, handles, co3, std_slope, ...
        std_intercept, Dose, mean_R2_ROI, std_error_mean_R2, gof_plot3);
    set(gca                         , ...
     'XColor'      , [.7 .7 .7] , ...
     'YColor'      , [.7 .7 .7] );
    hTitle  = title ('Linear regression of R2 vs. Dose');
```

```
    set( hTitle, 'Color', [.7 .7 .7]);

% Sets properties of figures and radio buttons.
    set(allchild(handles.axes1),'visible','off');
    set(handles.axes1,'visible','off');

    set(allchild(handles.axes2),'visible','on');
    set(handles.axes2,'visible','on');

    set(allchild(handles.axes3),'visible','off');
    set(handles.axes3,'visible','off');

    set(handles.radiobutton2,'Value',1)


% Code for closing waitbar
pause(.2)
set(handles.LoadingPanel,'visible','off')
set(handles.text_LoadingText,'string','')
set(handles.text_LoadingBar,'string','')



% --- Executes during object creation, after setting all properties.
function CalibrationPoints_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit1_Callback(hObject, eventdata, handles)

edit_textbox(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   % set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit4_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit10_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit14_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit15_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function edit16_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit17_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit18_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit19_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit20_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit21_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit22_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit23_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit24_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit25_Callback(hObject, eventdata, handles)
edit_textbox(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function uibuttongroup_Callback(hObject, eventdata, handles)


function edit1_ButtonDownFcn(hObject, eventdata, handles)

str2num(get(handles.edit1,'String'))

% --- Executes on button press in pushbutton_exportDir.
function pushbutton_exportDir_Callback(hObject, eventdata, handles)

global exportDir
exportDirTemp=[uigetdir,'\'];
if exportDirTemp ~ 0;
exportDir=exportDirTemp;
    set(handles.text_exportDir,'string',exportDir)
end
```

```
% --- Executes on button press in pushbutton_export.
function pushbutton_export_Callback(hObject, eventdata, handles)

global exportDir dir fileNames co2 co3 gof_plot2 gof_plot3 weight ...
    EchoTime pixelValueROI mean_R2_ROI Dose std_error_mean_R2 std_slope std_intercept

set(handles.text_exporting,'visible','on')
set(handles.text_exporting,'string','Exporting ...')

if exportDir ~ 0;
    filePathExport=exportDir;
else
    filePathExport=dir;
end

% Create a new folder with time stamp
    date_time=fix(clock);
    filePathExport=([filePathExport,'CalibData_',num2str(date_time(1)),'-',...
        sprintf('%02.0f',date_time(2)),'-',...
        sprintf('%02.0f',date_time(3)),'-',...
        sprintf('%02.0f',date_time(4)),...
        sprintf('%02.0f',date_time(5)),...
        sprintf('%02.0f',date_time(6)),'\']);
    mkdir(filePathExport);

% Code for .txt export
    fName=([filePathExport,'Data.txt']);

    str= [...
        sprintf(['Exported %d-%02.0f-%02.0f %02.0f:%02.0f:%02.0f \r\n\r\n'],...
            date_time(1),date_time(2),date_time(3),date_time(4),...
            date_time(5),date_time(6))...
        sprintf(['%d DICOM files selected:'],length(fileNames))...
        sprintf(['\r\n\t %s \r\n'],dir)...
        sprintf(['\t\t %s \r\n'],fileNames{:})...
        sprintf(['Echo times (ms): '])...
        sprintf(['%d '],EchoTime)...
        sprintf(['\r\n\r\n\r\n\r\nPlot 1 \r\n'...
        sprintf(['\r\nx-data (Echo time(s)): '])...
        sprintf(['%d '],EchoTime/1000)...
        sprintf(['\r\ny-data (mean pixel values of ROI #1 for all selected DICOM files): '])...
        sprintf(['%d '],mean(pixelValueROI{1}))...
        sprintf(['\r\nStandard deviation:  '])...
        sprintf(['%d '],std(pixelValueROI{1,1}))...
            '\r\nGeneral fit model: y(x) = S_0*exp(-R2*x) \r\n'...
            'Method: Weighted Nonlinear Least Squares \r\n'...
            'Algorithm: Levenberg-Marquardt \r\n'...
            '\t Coefficients: \r\n'...
            '\t\t R2=%d \r\n'...
            '\t\t S_0=%d \r\n'...
            'Goodness of fit: R^2 = %d \r\n'],co2(1),co2(2),gof_plot2.rsquare)...
        sprintf(['Weights (1/EchoTime)(s^-1): '])...
        sprintf(['%d '],weight)...
        sprintf(['\r\n\r\n\r\n\r\nPlot 2 \r\n'...
        sprintf(['\r\nx-data (absorbed dose (Gy)): '])...
        sprintf(['%d '],Dose)...
        sprintf(['\r\ny-data (mean calculated R2 values of ROIs): '])...
        sprintf(['%d '],mean_R2_ROI)...
        sprintf(['\r\nStandard error:  '])...
        sprintf(['%d '],std_error_mean_R2)...
            '\r\nGeneral fit model: y(x) = ax + b \r\n'...
            'Method: Linear Least Squares \r\n'...
            '\t Coefficients: \r\n'...
```

```
            '\t\t x=%d \r\n'...
            '\t\t b=%d \r\n'...
            'Goodness of fit: R^2 = %d \r\n'],co3(1),co3(2),gof_plot3.rsquare)...
            sprintf(['Standard deviation of slope: %d \r\n'],std_slope)...
            sprintf(['Standard deviation of intercept: %d'],std_intercept)...
        ];

    fid = fopen(fName,'w');       % Open/create file
    fprintf(fid,'%s\r\n',str);    % Print the string
    fclose(fid);                  % Close file


% Code for export of figures.
    fig=figure;
    set(fig, 'visible','off')
    plot_decay(hObject, eventdata, handles, co2, gof_plot2, EchoTime, pixelValueROI, Dose);
    saveas(fig,[filePathExport 'DecayCurve.png'],'png')
    set(fig, 'visible','on')
    close(fig)

    set(handles.text_exporting,'string','Exporting ....')

    fig=figure;
    set(fig, 'visible','off')
    plot_decay_NoText(hObject, eventdata, handles, co2, gof_plot2, EchoTime, pixelValueROI);
    saveas(fig,[filePathExport 'DecayCurve_NoText.png'],'png')
    set(fig, 'visible','on')
    close(fig)

    set(handles.text_exporting,'string','Exporting .....')

    fig=figure;
    set(fig, 'visible','off')
    plot_linear(hObject, eventdata, handles, co3, std_slope, ...
        std_intercept, Dose, mean_R2_ROI, std_error_mean_R2, gof_plot3);
    saveas(fig,[filePathExport 'R2vsDose.png'],'png')
    set(fig, 'visible','on')
    close(fig)

    set(handles.text_exporting,'string','Exporting ......')

    fig=figure;
    set(fig, 'visible','off')
    plot_linear_NoText(hObject, eventdata, handles, co3, std_slope, ...
        std_intercept, Dose, mean_R2_ROI, std_error_mean_R2, gof_plot3);
    saveas(fig,[filePathExport 'R2vsDose_NoText.png'],'png')
    set(fig, 'visible','on')
    close(fig)

set(handles.text_exporting,'visible','off')
set(handles.text_exportSuccess,'visible','on')


function edit_textbox(hObject, eventdata, handles)

global Dose NumberOfROIs flag_DoneROI flag_NotCalculate flag_breakCalculate

% Fills "Dose" with the values of active text boxes. If no string is present
% in a text box, NaN is written in its place. Checks if there are any NaN
% in the vector "Dose", if so, sets flag to 1.
handlesStructure = guihandles(gcf);
flag_NotCalculate=0;
flag_breakCalculate=1;
for i= 1:NumberOfROIs
    h = handlesStructure.(sprintf('edit%d', i));
```

```
    Dose(i)= str2double(get(h, 'String'));
    if isnan(Dose(i))== 1
       flag_NotCalculate=1;
    end
end

if flag_NotCalculate == 0 && length(Dose)== NumberOfROIs && flag_DoneROI==1
    set(handles.text_SpecifyDose,'Visible','Off')
    set(handles.pushbutton_Calculate,'Enable','On')
    flag_breakCalculate = 0;

elseif flag_DoneROI==1
    set(handles.text_SpecifyDose,'Visible','On')
    set(handles.pushbutton_Calculate,'Enable','Off')
else
    set(handles.pushbutton_Calculate,'Enable','Off')
end

% Makes pushbutton_Calculate pulse
while flag_breakCalculate == 0
        for i = 20:-1:1
            i=(i/150);
            i=[.702-.05+i .710-.05+i .796-.05+i];
            set(handles.pushbutton_Calculate,'BackgroundColor',i)
            pause(.035)
        end
        for i = 1:20
            i=(i/150);
            i=[.702-.05+i .710-.05+i .796-.05+i];
            set(handles.pushbutton_Calculate,'BackgroundColor',i)
            pause(.030)
        end
end

end


function loading(hObject, eventdata, handles, perc)

    step=floor(perc/2);
    str=sprintf('Calculating... %d%%', perc);
    set(handles.text_LoadingText,'string',str)
    set(handles.LoadingPanel,'visible','on')

    str='';
    for i=1:step
        str=[sprintf('%s', str,'-')];
        set(handles.text_LoadingBar,'string',str)
        set(handles.text_LoadingBar,'foregroundcolor',[.694-i/200 .700+i/170 .804-i/200])

    end
end


function plot_decay(hObject, eventdata, handles, co2, gof_plot2, EchoTime, pixelValueROI, Dose)

    % x from 0 to the x corresponding ~4% of signal of S_0
    x=linspace(0,round(log(.04)/(-co2(1))*10)/10);

    hold on;
    hFit   = line(x,co2(2)*exp(-co2(1)*x));
    ErrLength=0.04;
    y= mean(pixelValueROI{1});
    std_err=std(pixelValueROI{1,1});
    for i = 1:length(EchoTime)
```

```
    plot([EchoTime(i)/1000-ErrLength EchoTime(i)/1000+ErrLength],[y(i)-std_err(i),y(i)-std_err(i)],
    plot([EchoTime(i)/1000-ErrLength EchoTime(i)/1000+ErrLength],[y(i)+std_err(i),y(i)+std_err(i)],
end
hE     = errorbar(EchoTime/1000, mean(pixelValueROI{1}),std(pixelValueROI{1,1}));

set(hFit                              , ...
    'Color'         , [0 0 .5]        , ...
    'LineWidth'     , 2               );

set(hE                                , ...
  'LineStyle'       , 'none'          , ...
  'Color'           , [.25 .25 .25]   , ...
  'LineWidth'       , 1               , ...
  'Marker'          , 'o'             , ...
  'MarkerSize'      , 8               , ...
  'MarkerEdgeColor' , [.25 .25 .25]   , ...
  'MarkerFaceColor' , [.7 .7 .7]      );

hLegend = legend( ...
  [hE, hFit], ...
  'Data (\mu \pm \sigma)' , ...
  'Fit, y(x) = S_o\cdote^{-R2\cdotx}'       , ...
  'location', 'NorthWest' );

 set(gca                        , ...
 'YGrid'       , 'on'           , ...
 'XGrid'       , 'on'           , ...
 'FontName'    , 'Serif'        , ...
 'Box'         , 'off'          , ...
 'TickDir'     , 'out'          , ...
 'TickLength'  , [.015 .015]    , ...
 'XMinorTick'  , 'off'          , ...
 'YMinorTick'  , 'off'          , ...
 'YGrid'       , 'on'           , ...
 'XColor'      , [.3 .3 .3]     , ...
 'YColor'      , [.3 .3 .3]     , ...
 'LineWidth'   , 1              );

hTitle  = title ('Exponential fit of data values of ROI selection 1');
hXLabel = xlabel('Echo time, TE (s)');
hYLabel = ylabel('Relative pixel intensity');

set([hTitle, hXLabel, hYLabel] , ...
    'FontName'   , 'Serif');
set([hLegend, gca]             , ...
    'FontSize'   , 11          );
set([hXLabel, hYLabel]         , ...
    'FontSize'   , 14          );
set( hTitle                    , ...
    'FontSize'   , 14          , ...
    'FontWeight' , 'bold'      );

text('Position',[max(x)/10*5.5 co2(2)/10*7.8],'String', ...
    {['Absorbed dose in ROI selection 1 = ',num2str(Dose(1)),' Gy'];...
    'General fit model: y(x) = S_0\cdote^{-R2\cdotx}'; ...
   'Method: Weighted Nonlinear Least Squares'; ...
   'Algorithm: Levenberg-Marquardt'; ...
   ''; ...
   '                   Coefficients:'; ...
   ['                       R2 = ',num2str(co2(1))]; ...
   ['                       S_0  = ',num2str(co2(2))]; ...
   ''; ...
   ['Goodness of fit:   R^2 = ',num2str(gof_plot2.rsquare)]})

xlim([0 max(x)])
```

```
end


function plot_decay_NoText(hObject, eventdata, handles, co2, gof_plot2, EchoTime, pixelValueROI)

    % x from 0 to the x corresponding ~4% of signal of S_0
    x=linspace(0,round(log(.04)/(-co2(1))*10)/10);

    hold on;
    hFit   = line(x,co2(2)*exp(-co2(1)*x));
    ErrLength=0.04;
    y= mean(pixelValueROI{1});
    std_err=std(pixelValueROI{1,1});
    for i = 1:length(EchoTime)
        plot([EchoTime(i)/1000-ErrLength EchoTime(i)/1000+ErrLength],[y(i)-std_err(i),y(i)-std_err(i)],'k',
        plot([EchoTime(i)/1000-ErrLength EchoTime(i)/1000+ErrLength],[y(i)+std_err(i),y(i)+std_err(i)],'k',
    end
    hE     = errorbar(EchoTime/1000, mean(pixelValueROI{1}),std(pixelValueROI{1,1}));

    set(hFit                              , ...
        'Color'         , [0 0 .5]      , ...
        'LineWidth'     , 2             );

    set(hE                                , ...
      'LineStyle'       , 'none'        , ...
      'Color'           , [.25 .25 .25] , ...
      'LineWidth'       , 1             , ...
      'Marker'          , 'o'           , ...
      'MarkerSize'      , 8             , ...
      'MarkerEdgeColor' , [.25 .25 .25] , ...
      'MarkerFaceColor' , [.7 .7 .7]    );

    hLegend = legend( ...
      [hE, hFit], ...
      'Data (\mu \pm \sigma)' , ...
      'Fit, y(x) = S_o\cdote^{-R2\cdotx}'      , ...
      'location', 'NorthWest' );

     set(gca                       , ...
     'YGrid'       , 'on'        , ...
     'XGrid'       , 'on'        , ...
     'FontName'    , 'Serif'     , ...
     'Box'         , 'off'       , ...
     'TickDir'     , 'out'       , ...
     'TickLength'  , [.015 .015], ...
     'XMinorTick'  , 'off'       , ...
     'YMinorTick'  , 'off'       , ...
     'YGrid'       , 'on'        , ...
     'XColor'      , [.3 .3 .3] , ...
     'YColor'      , [.3 .3 .3] , ...
     'LineWidth'   , 1           );

    hTitle  = title ('Exponential fit of data values of ROI selection 1');
    hXLabel = xlabel('Echo time, TE (s)');
    hYLabel = ylabel('Relative pixel intensity');

    set([hTitle, hXLabel, hYLabel] , ...
        'FontName'   , 'Serif');
    set([hLegend, gca]             , ...
        'FontSize'   , 11          );
    set([hXLabel, hYLabel]         , ...
        'FontSize'   , 14          );
    set( hTitle                    , ...
        'FontSize'   , 14          , ...
        'FontWeight' , 'bold'      );
```

```
   xlim([0 max(x)])
end


function plot_linear(hObject, eventdata, handles, co3, std_slope, std_intercept, Dose, mean_R2_ROI, std

x=[0 max(Dose)+1];

hFit   = line(x,co3(1)*x + co3(2));
hold on

% Plot standard deviation of data points, the slope and intercept
if std_slope >= 0
           hCI(1) = line(x,(co3(1)-std_slope)*x + co3(2)-std_intercept);
           hCI(2) = line(x,(co3(1)+std_slope)*x + co3(2)+std_intercept);
   else
           hCI(1)= line(x,(co3(1)-std_slope)*x + co3(2)+std_intercept);
           hCI(2)= line(x,(co3(1)+std_slope)*x + co3(2)-std_intercept);
end

ErrLength=0.20;
for i = 1:length(Dose)
    plot([Dose(i)-ErrLength Dose(i)+ErrLength],[mean_R2_ROI(i)-std_error_mean_R2(i),mean_R2_ROI(i)-std_e
    plot([Dose(i)-ErrLength Dose(i)+ErrLength],[mean_R2_ROI(i)+std_error_mean_R2(i),mean_R2_ROI(i)+std_e
end
hE     = errorbar(Dose,mean_R2_ROI,std_error_mean_R2,'.k');

    set(hFit                              , ...
        'Color'         , [0 0 .5]      , ...
        'LineWidth'     , 2             );

    set(hE                                , ...
      'LineStyle'       , 'none'        , ...
      'Color'           , [.25 .25 .25] , ...
      'LineWidth'       , 1             , ...
      'Marker'          , 'o'           , ...
      'MarkerSize'      , 8             , ...
      'MarkerEdgeColor' , [.25 .25 .25] , ...
      'MarkerFaceColor' , [.7 .7 .7]    );

     set(hCI(1)                           , ...
      'LineStyle'       , '--'          , ...
      'LineWidth'       , 1.5           , ...
      'Color'           , [0 .4 0]      );
     set(hCI(2)                           , ...
      'LineStyle'       , '--'          , ...
      'LineWidth'       , 1.5           , ...
      'Color'           , [0 .4 0]      );

    hLegend = legend( ...
    [hE, hFit, hCI(1)], ...
    'Data (\mu \pm \sigma)'         , ...
    'Fit, y(x) = ax + b'           , ...
    '68% (\sigma) CI'               , ...
    'location', 'NorthWest'         );

    set(gca                       , ...
      'YGrid'       , 'on'        , ...
      'XGrid'       , 'on'        , ...
      'FontName'    , 'Serif'     , ...
      'Box'         , 'off'       , ...
      'TickDir'     , 'out'       , ...
      'TickLength'  , [.015 .015], ...
      'XMinorTick'  , 'off'       , ...
```

```
     'YMinorTick'  , 'off'      , ...
     'YGrid'       , 'on'       , ...
     'XColor'      , [.3 .3 .3] , ...
     'YColor'      , [.3 .3 .3] , ...
     'LineWidth'   , 1          );

    hTitle  = title ('Linear regression of R2 vs. Dose');
    hXLabel = xlabel('Dose (Gy)');
    hYLabel = ylabel('R2 (s^{-1})');

    set([hTitle, hXLabel, hYLabel] , ...
        'FontName'   , 'Serif');
    set([hLegend, gca]             , ...
        'FontSize'   , 11          );
    set([hXLabel, hYLabel]         , ...
        'FontSize'   , 14          );
    set( hTitle                    , ...
        'FontSize'   , 14          , ...
        'FontWeight' , 'bold'      );

    text('FontName'   , 'Serif', ...
        'Position',[(max(x)/10*6.5) ...
        (((max(co3(1)*x + co3(2))-co3(2))/10*3)+co3(2))],'String', ...
        {'General fit model: y(x) = ax + b'; ...
        'Method: Linear Least Squares'; ...
        ''; ...
        '                    Coefficients:'; ...
        ['                        x = ',num2str(co3(1))]; ...
        ['                        b = ',num2str(co3(2))]; ...
        ''; ...
        ['Goodness of fit:   R^2 = ',num2str(gof_plot3.rsquare)]})

    xlim([0 max(x)])
end


function plot_linear_NoText(hObject, eventdata, handles, co3, std_slope, std_intercept, Dose, mean_R2_ROI,

x=[0 max(Dose)+1];

hFit   = line(x,co3(1)*x + co3(2));
hold on

% Plot standard deviation of data points, the slope and intercept
if std_slope >= 0
          hCI(1) = line(x,(co3(1)-std_slope)*x + co3(2)-std_intercept);
          hCI(2) = line(x,(co3(1)+std_slope)*x + co3(2)+std_intercept);
   else
          hCI(1)= line(x,(co3(1)-std_slope)*x + co3(2)+std_intercept);
          hCI(2)= line(x,(co3(1)+std_slope)*x + co3(2)-std_intercept);
end

ErrLength=0.20;
for i = 1:length(Dose)
    plot([Dose(i)-ErrLength Dose(i)+ErrLength],[mean_R2_ROI(i)-std_error_mean_R2(i),mean_R2_ROI(i)-std_erro
    plot([Dose(i)-ErrLength Dose(i)+ErrLength],[mean_R2_ROI(i)+std_error_mean_R2(i),mean_R2_ROI(i)+std_erro
end

hE     = errorbar(Dose,mean_R2_ROI,std_error_mean_R2,'.k');

    set(hFit                       , ...
        'Color'         , [0 0 .5]     , ...
        'LineWidth'     , 2            );

    set(hE                         , ...
```

```
      'LineStyle'       , 'none'        , ...
      'Color'           , [.25 .25 .25] , ...
      'LineWidth'       , 1             , ...
      'Marker'          , 'o'           , ...
      'MarkerSize'      , 8             , ...
      'MarkerEdgeColor' , [.25 .25 .25] , ...
      'MarkerFaceColor' , [.7 .7 .7]    );

     set(hCI(1)                         , ...
      'LineStyle'       , '--'          , ...
      'LineWidth'       , 1.5           , ...
      'Color'           , [0 .4 0]      );
     set(hCI(2)                         , ...
      'LineStyle'       , '--'          , ...
      'LineWidth'       , 1.5           , ...
      'Color'           , [0 .4 0]      );

    hLegend = legend( ...
    [hE, hFit, hCI(1)], ...
    'Data (\mu \pm \sigma)'         , ...
    'Fit, y(x) = ax + b'           , ...
    '68% (\sigma) CI'               , ...
    'location', 'NorthWest'         );

    set(gca                       , ...
      'YGrid'       , 'on'        , ...
      'XGrid'       , 'on'        , ...
      'FontName'    , 'Serif'     , ...
      'Box'         , 'off'       , ...
      'TickDir'     , 'out'       , ...
      'TickLength'  , [.015 .015], ...
      'XMinorTick'  , 'off'       , ...
      'YMinorTick'  , 'off'       , ...
      'YGrid'       , 'on'        , ...
      'XColor'      , [.3 .3 .3] , ...
      'YColor'      , [.3 .3 .3] , ...
      'LineWidth'   , 1           );

    hTitle  = title ('Linear regression of R2 vs. Dose');
    hXLabel = xlabel('Dose (Gy)');
    hYLabel = ylabel('R2 (s^{-1})');

    set([hTitle, hXLabel, hYLabel] , ...
        'FontName'   , 'Serif');
    set([hLegend, gca]             , ...
        'FontSize'   , 11          );
    set([hXLabel, hYLabel]         , ...
        'FontSize'   , 14          );
    set( hTitle                    , ...
        'FontSize'   , 14          , ...
        'FontWeight' , 'bold'      );

    xlim([0 max(x)])
end
```