09/11/2018

Scarlett Royle, 201245875

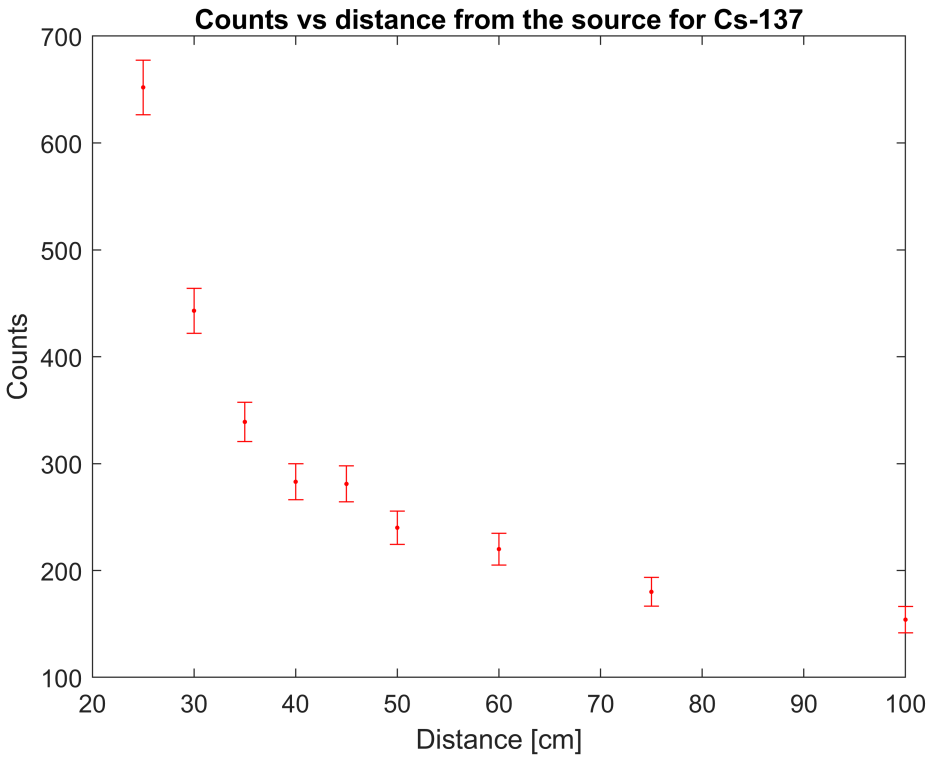
PHYS205: Problem Set 4

Question 1

In this question, Matlab fitting functions were used on the counts vs distance data collected from a Cs-137 source to verify the $\frac{1}{r^2}$ relationship for counting.

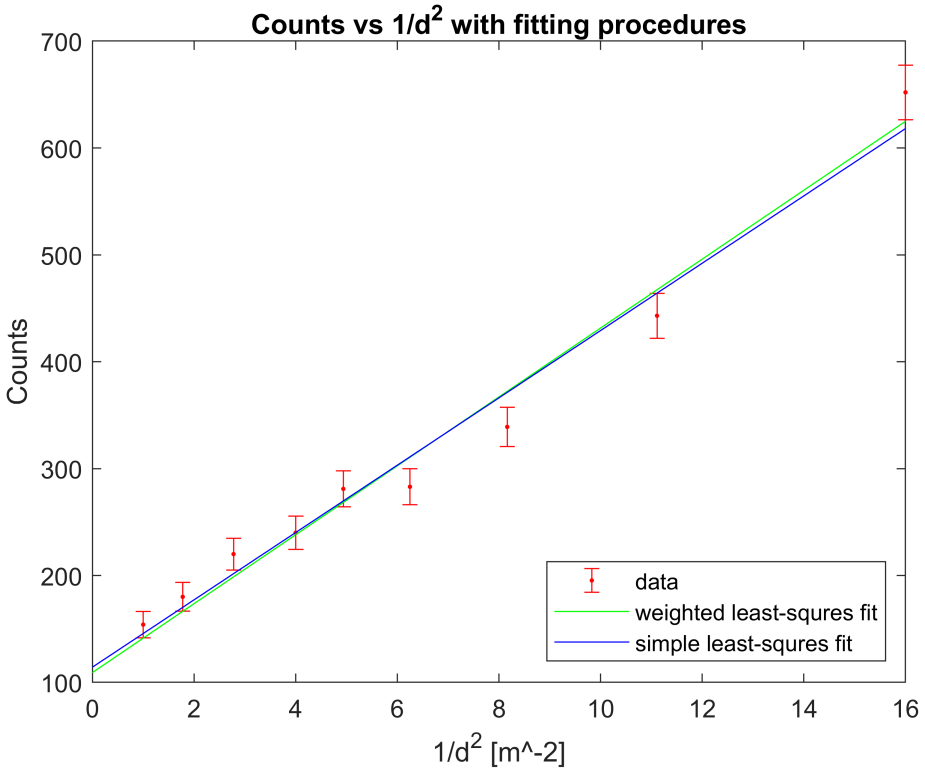
Firstly the raw data were plotted, then counts vs $\frac{1}{r^2}$. This plot was fitted by using both a simple least-squares fit and a weighted least-squares fit. The $\frac{\chi^2}{\mathrm{NDF}}$ values calculated for each were 1.66 and 1.88 respectively, suggesting that, for these data, a simple least-squares fit is most appropriate. Both values are between 0.25 and 4, showing that there is indeed a $\frac{1}{r^2}$ relationship for counting.

```
%Plotting the data:
counting_data = csvread('counts_vs_distance.csv', 1,0); %loading the Cs counts (column 2)
% vs distance (column 1) data
d = counting_data(:,1); %distance from the souce
counts = counting_data(:,2); %number of counts
d_counts = sqrt(counts); %the uncertainty on the counts = sqrt(counts)
n = length(d); %number of data points

errorbar(d, counts, d_counts, '.r'); %plotting the data on figure(1) with errors in counts
% (y-axis)
xlabel('Distance [cm]');
ylabel('Counts');
title('Counts vs distance from the source for Cs-137');
```



Plot 1: Counts detected vs distance for a Cs-137 source.

```
%Fitting an appropriate function to the data:
for i = 1:n
    fit_d(i,1) = 1/(d(i,1)*10^-2)^2; %1/d^2 where d is in m. Seeing whether the data
    % agrees with the 1/r^2 relationship for counting
end

[least_squares, ~] = fit(fit_d, counts, 'poly1', 'weight', d_counts); %weighted least
% squares fit
[simple_fit, ~] = fit(fit_d, counts, 'poly1'); %simple least squares fit

errorbar(fit_d, counts, d_counts, '.r'); %plotting counts vs 1/d^2 with errorbars in counts
hold on
plot(least_squares, '-g'); %plotting the weighted least-squares fit
plot(simple_fit, '-b'); %plotting the simple least-squares fit
xlabel('1/d^2 [m\^-2]');
ylabel('Counts');
title('Counts vs 1/d^2 with fitting procedures');
legend('data', 'weighted least-squres fit', 'simple least-squres fit', 'location', ...
    'southeast');
hold off
```



Plot 2: Counts vs 1/d^2 with both a simple least-squares fit (blue) and a weighted-least squares fit (green) plotted.

Upon visual examination, there is indeed a linear relationship between the counts and $\frac{1}{d^2}$, and both fits fit the data well. The simple least-squares fit seems to fit the data better as it is within more of the error bars.

Next, the $\frac{\chi^2}{\text{NDF}}$ values were calculated for each fit using a function defined in *Appendix 1* to reach a more informed conclusion:

```
%Finding the ChiSqr/NDF for each fitting technique using the function
%defined in Appendix 1:
[~,~,~, ChiSqr_NDF_ls] = chiSqr(fit_d, counts, d_counts, least_squares, 2);
%for simple least-squares
[~,~,~, ChiSqr_NDF_sf] = chiSqr(fit_d, counts, d_counts, simple_fit, 2);
%for weighted least-squares

fit_type = {'Weighted Least-squares'; 'Simple Least-squares'};
ChiSqr_NDF = [ ChiSqr_NDF_ls ; ChiSqr_NDF_sf ]; %column vector of the
% ChiSqr/NDF values for each fitting technique
fit_results = table(fit_type, ChiSqr_NDF);
disp(fit_results)
```

```
          fit_type            ChiSqr_NDF
    ____________________________    __________

    'Weighted Least-squares'        1.8765
    'Simple Least-squares'          1.6556
```

Table 1: The $\frac{\chi^2}{\mathrm{NDF}}$ values for each fitting procedure.

Both the $\frac{\chi^2}{\mathrm{NDF}}$ values are between 0.25 and 4, suggesting a good fit and that errors are well estimated.

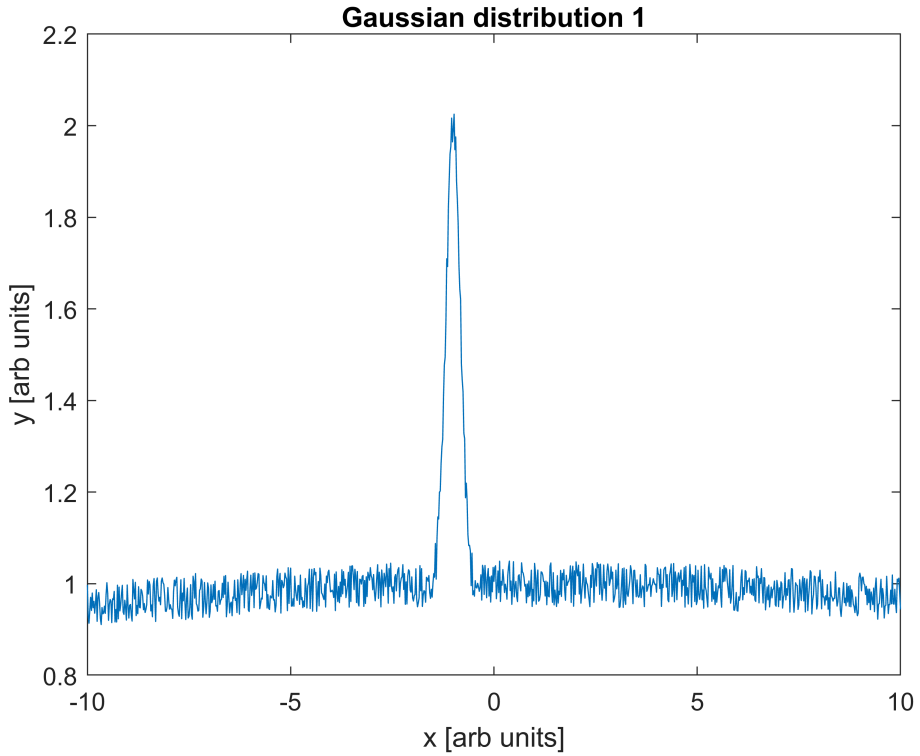
The $\frac{\chi^2}{\mathrm{NDF}}$ for the simple fit is closer to 1 than that for the weighted fit. Therefore, the simple fit is better for this data.

Question 2

The Gaussian function is given by $f(x) = ae^{\frac{(x-b)^2}{2c^2}}$. By varying the parameters *a*, *b* and *c*, the properties of the distribution change. In this question, the parameters of a covolved Gaussian distribution are varied and the effect on the shape of the distribution is observed using the Curve Fitting toolbox.

The given code was used to produce a covolved Gaussian distribution from the sum of two distributions with different parameters:

```
a1 = 1; b1 = -1; c1 = 0.25;
a2 = 1; b2 = 1; c2 = 50;
x1 = (-10:0.02:10);
gdata = a1*exp(-((x1-b1)/c1).^2) + a2*exp(-((x1-b2)/c2).^2) + 0.1*(rand(size(x1))-.5);
error_gdata = sqrt(gdata);
plot(x1,gdata)
title('Gaussian distribution 1');
xlabel('x [arb units]');
ylabel('y [arb units]');
```



Plot 3: Covolved Gaussian distribution with set parameters a1 = 1, b1 = -1, c1 = 0.25, a2 = 1, b2 = 1, and c2 = 50.

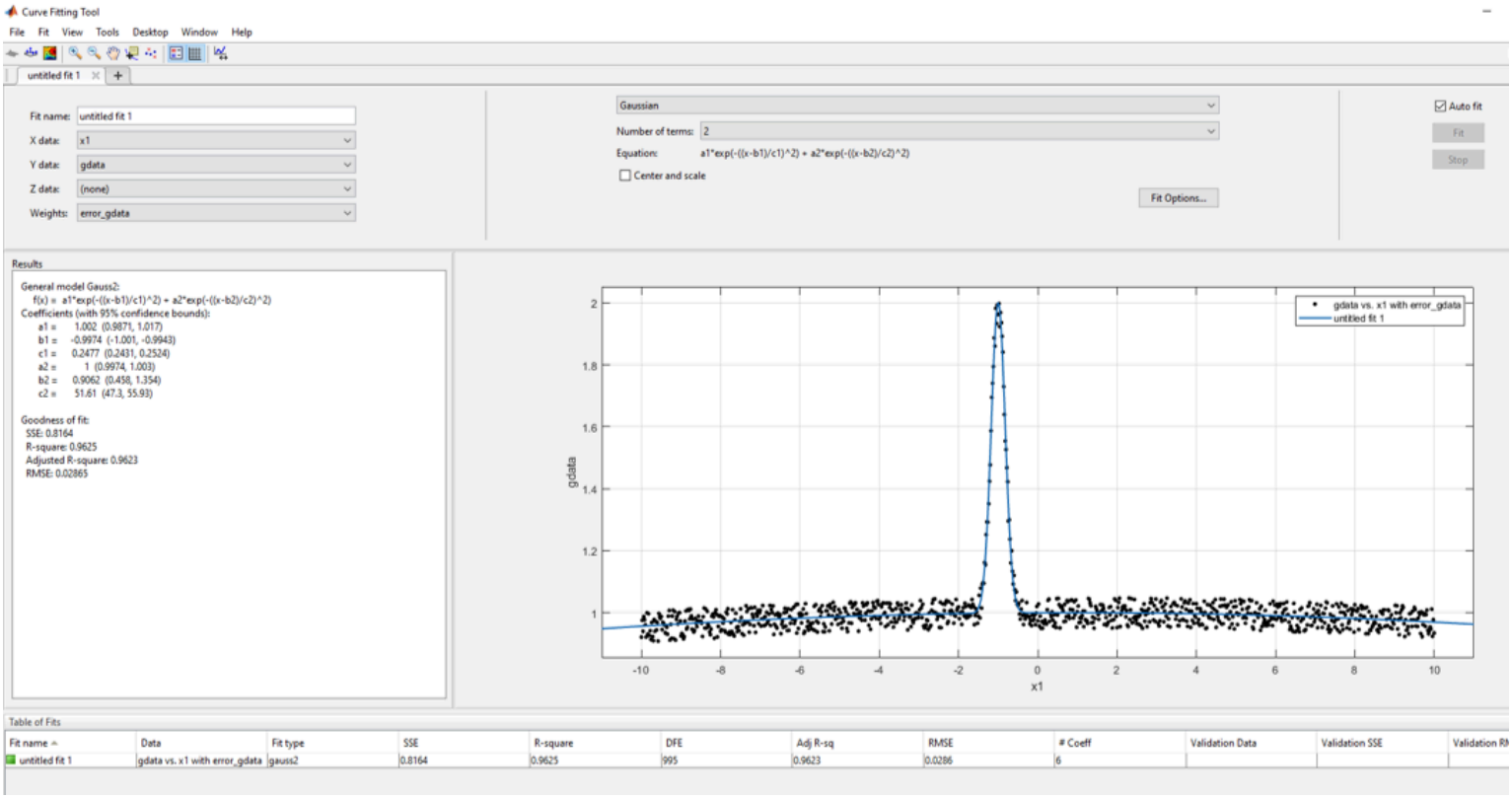


Image 1: The Gausian distribution with set parameters a1 = 1, b1 = -1, c1 = 0.25, a2 = 1, b2 = 1, and c2 = 50 fitted via the Curve Fitting toolbox. The fit parameters are displayed on the right hand side of the image.

The parameters given by the Curve Fitting toolbox can be seen on the right hand side of the *Image 1*, which were a1 = 1.002, b1 = -0.9974, c1 = 0.2477, a2 = 1, b2 = 0.9062 and c2 = 51.61.

The parameters entered into Matlab were then changed to a1 = 2, b1 = -2, c1 = 0.5, a2 = 2, b2 = 2, and c2 = 100 and the effect on the distribution was observed:

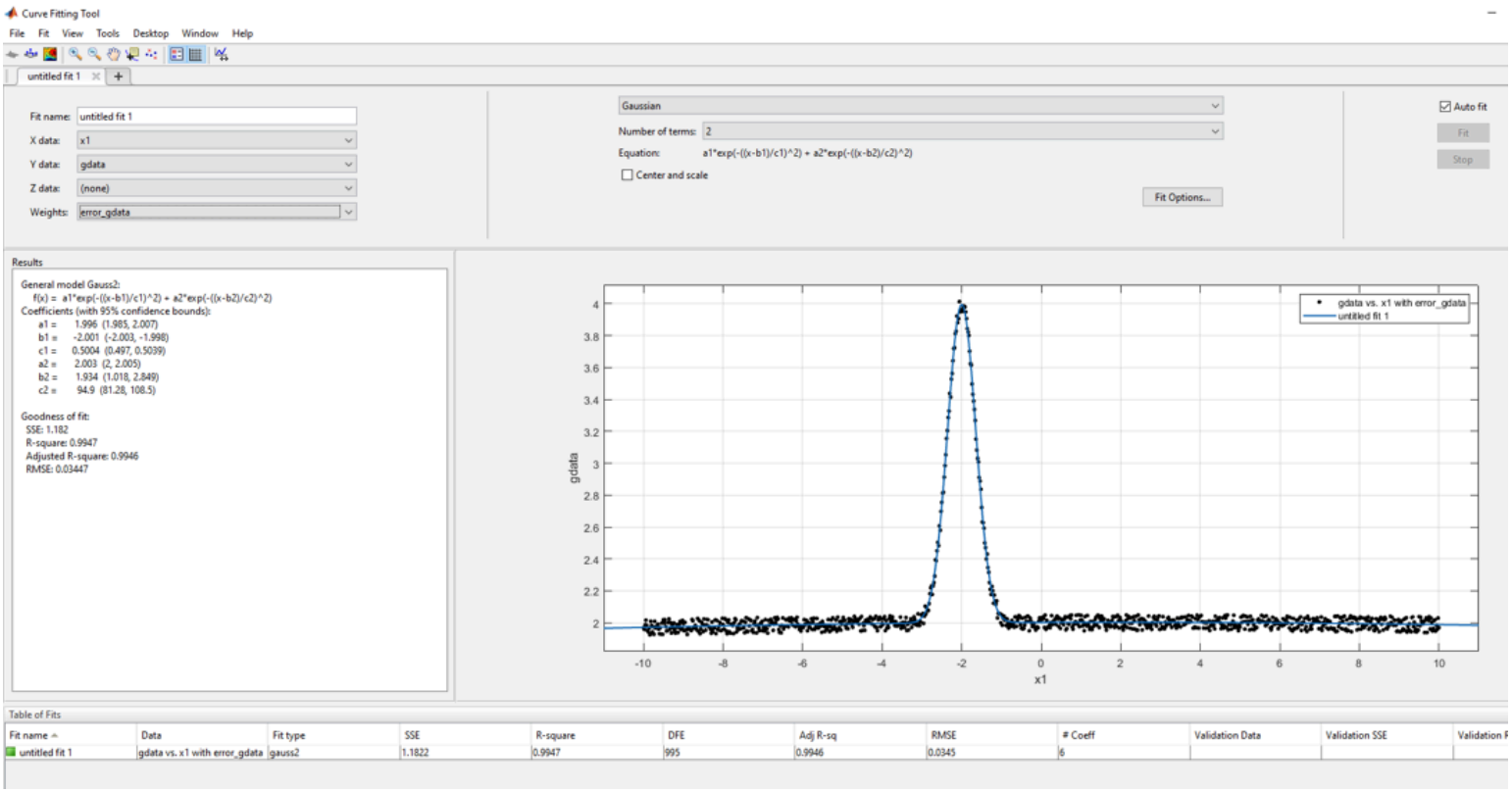


Image 2: the Gausian distribution with set parameters a1 = 2, b1 = -2, c1 = 0.5, a2 = 2, b2 = 2, and c2 = 100 fitted via the Curve Fitting toolbox. The fit parameters are displayed on the right hand side of the image.

Increasing the *a* parameters incresed the height of the peak, increasing the *b* parameters moved the plot to the right (up the x-axis) and increasing the *c* parameters increased the spread of distribution.

Changing one of *b1* or *b2* with respect to the other, made the plot appear more devolved; two peaks could be seen as one shifted to the right more than the other. Increasing one of *a1* or *a2* and also increasing the corresponding *c* variable also made the distribution more devolved. The distribution with the lower *a* and *c* values would produce a line with a smaller peak and a greater width, like a hump, which then had a shrper, greater peak emerging due to the summation with the higher *a2* and *c2* values. This can be seen in *Image 3* below:

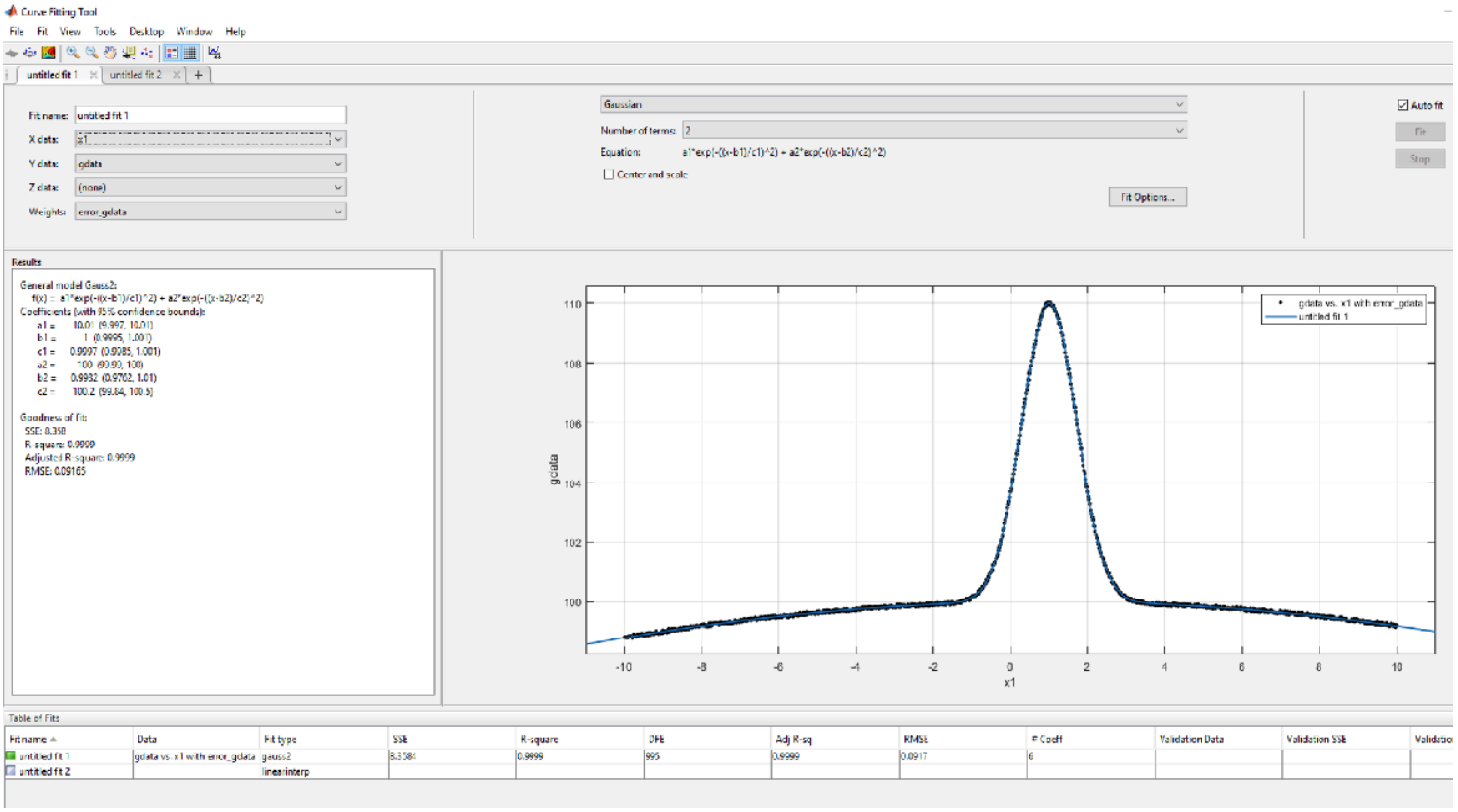


Image 3: The Gausian distribution with parameters set as a1 = 10, b1 = 1, c1 = 1, a2 = 1, b2 = 1 and c2 = 100 fitted via the Curve Fitting toolbox. Increasing both a1 and c1 relative to a2 and c2 gave a distribution which appears like a hump with a sharp peak emerging.

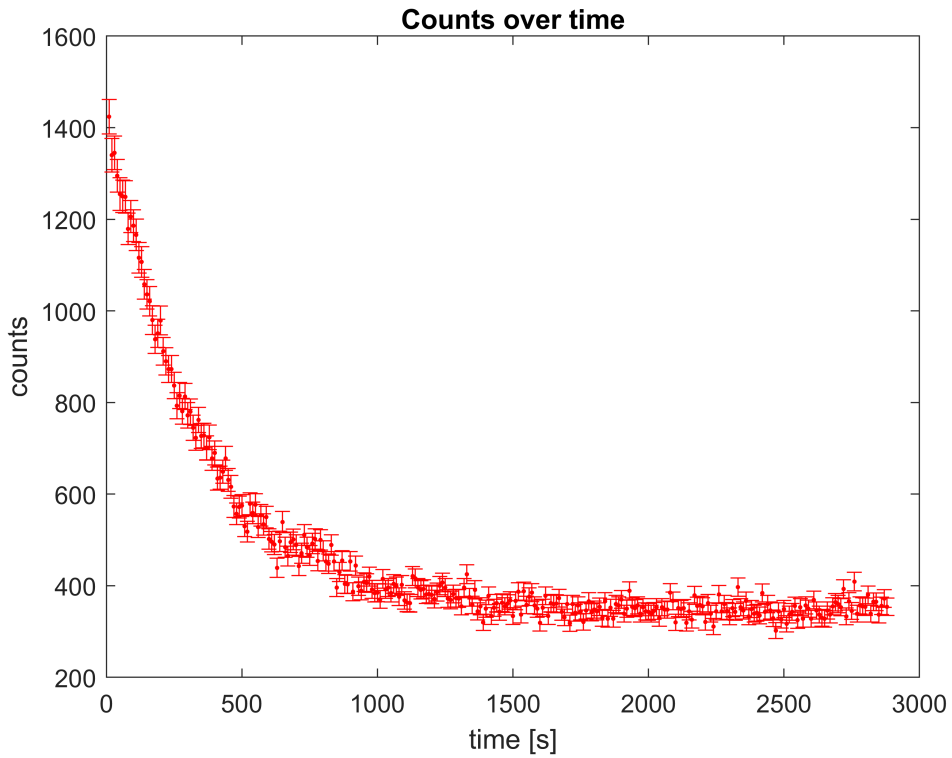
Question 3

In this question, a set of experimental data collected using a NaI(Tl) scintillation detector to record the counts over time from a neutron activated sample was fitted using two techniques: firstly, by exploiting an inbuilt Matlab fitting function and secondly, by employing an iterative approach to find the fit gradient which gave the smallest χ^2 value. Each gradient was used to calculate the decay constant, λ, for the sample as well as its half life, $T_{\frac{1}{2}}$. The results are summarised in *Table 2* at the end of this question.

3a: Loading the data and subtracting the background

In this section, the mean background count was found (349 ± 3 counts) and then subtracted from the data.

```
detector_data = csvread('problems4_raw_data.csv',1,0);
t = detector_data(:,1); %time points
counts2 = detector_data(:,2); %counts
n2 = length(t); %the number of data points
d_counts2 = sqrt(counts2); %uncertainty on the counts

figure(3)
errorbar(t, counts2, d_counts2, '.r')
xlabel('time [s]');
ylabel('counts');
title('Counts over time');
```



Plot 4: A plot of the counts detected using a NAI(Tl) scintillator vs time including the background count.

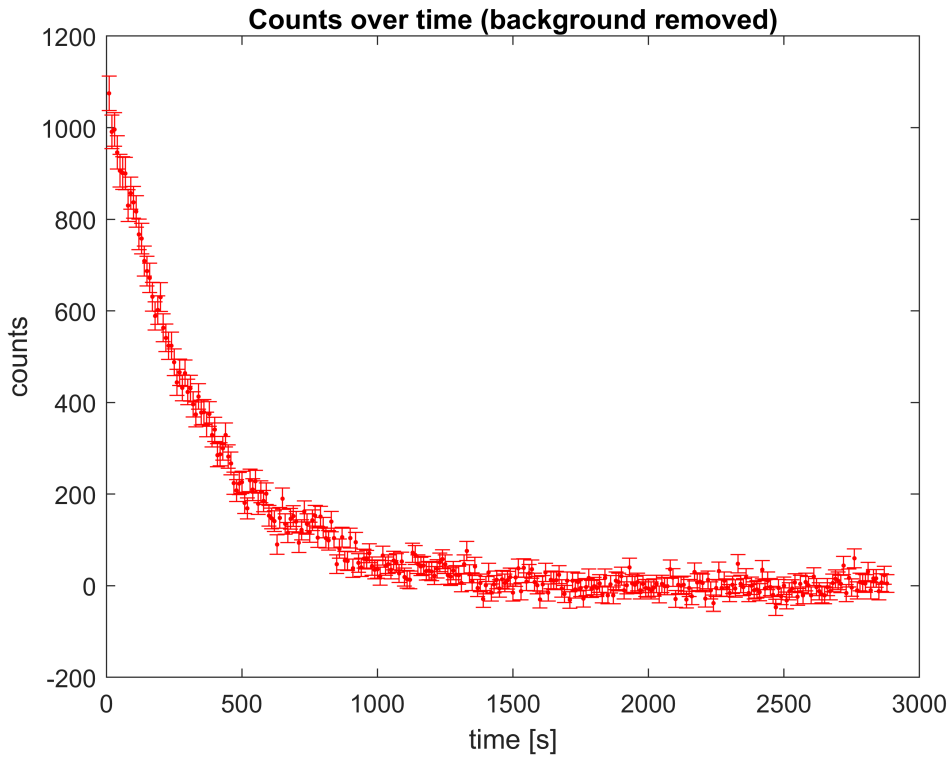
Next, the background was subtracted:

```
%Subtracting the background:
background = counts2(250:280); %chosing some points to be the background
mean_background = mean(background);%mean backgound count
d_background = std(background)/sqrt(length(background)); %error on the background

fprintf('The mean estimated background is %d and its error is %d\n', ...
    mean_background, d_background)
```

```
The mean estimated background is 349 and its error is 3.442305e+00
```

The mean background count was found to be 349 ± 3 counts. This was then subtracted from the data:

```
counts2_corrected = counts2 - mean_background; %counts with background removed
d_counts2_corrected = sqrt(d_counts2.^2 + d_background^2); %error on the counts with
% background removed

figure(4)
errorbar(t, counts2_corrected, d_counts2_corrected, '.r');
xlabel('time [s]');
ylabel('counts');
title('Counts over time (background removed)');
```



Plot 5: A plot of the counts detected using a NAI(Tl) scintillator vs time with the background count subtracted.

3c: Fitting the data using the standard Matlab fitting functions

Part 3c was conducted before part 3b so that the procedures involved in determining $T_{\frac{1}{2}}$ could be better understood before using an iterative approach.

Using

$$N = N_0 \cdot e^{-\lambda t},$$

one can derive a linear dependence between $\ln(N)$ and λ:

$$\ln(N) = -\lambda t + \ln(N_0).$$

Hence, a plot of $\ln(N)$ vs *t* shold produce a linear fit with the gradient equal to $-\lambda$. Using Matlab fitting functions, the data were plotted, fitted and λ extracted. The uncertainty in λ was equal to that for the gradient.

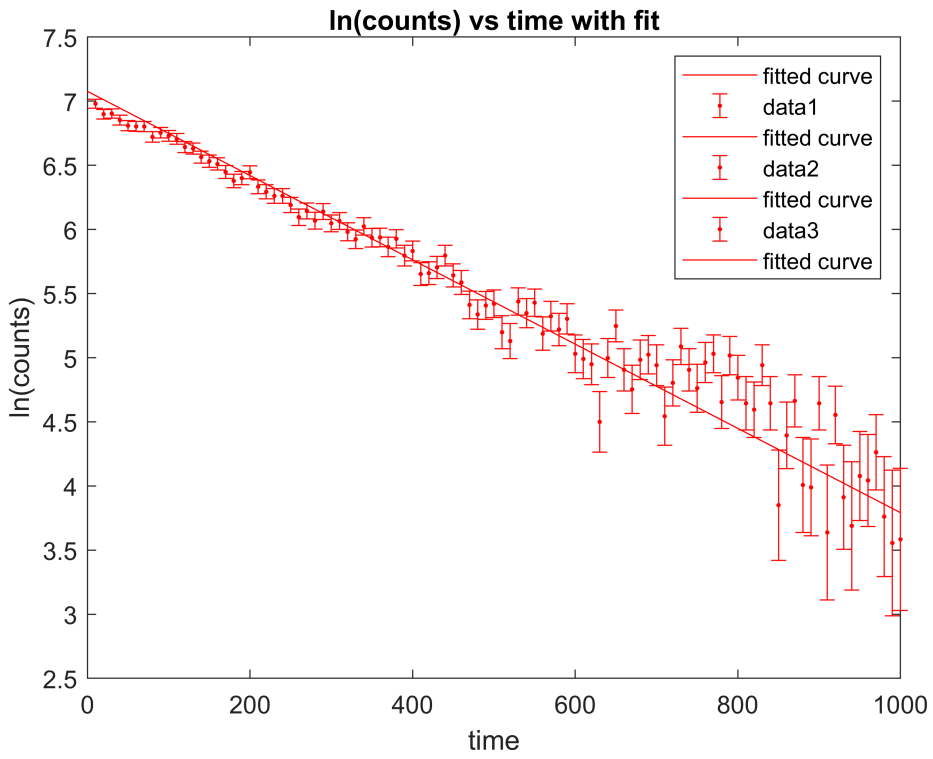
From λ, $T_{\frac{1}{2}}$ was calculated by setting $N = \frac{1}{2}N_0$:

$$T_{\frac{1}{2}} = \frac{\ln(2)}{\lambda}.$$

The uncertainty in $T_{\frac{1}{2}}$ was calculated by adding the uncertainties on the parameters in quadrature.

Using this non-iterative approach, $\lambda = 0.0033 \pm 0.0002\ s^{-1}$ and $T_{\frac{1}{2}} = 210 \pm 10\ s$ or $3.5 \pm 0.2\ \text{min}$. For the fitting procedure, $\chi^2 = 194$.

```
%Linear plot of ln(counts) vs d:
ln_counts2 = real(log(counts2_corrected)); %the ln of the counts corrected for background
%real(X) gives the complex real part of X
d_ln_counts2 = d_counts2_corrected ./ counts2_corrected; %the uncertainty in ln(counts)

t_short = t(1:100); %truncating the data
ln_counts2_short = ln_counts2(1:100);
d_ln_counts2_short = d_ln_counts2(1:100);

%3c
%non-iterative fit parameters:
[fit2, gof] = fit(t_short, ln_counts2_short, 'poly1', 'weight', d_ln_counts2_short);

figure(5)
errorbar(t_short, ln_counts2_short, d_ln_counts2_short, '.r')
hold on
plot(fit2);
title('ln(counts) vs time with fit');
xlabel('time');
ylabel('ln(counts)')
```

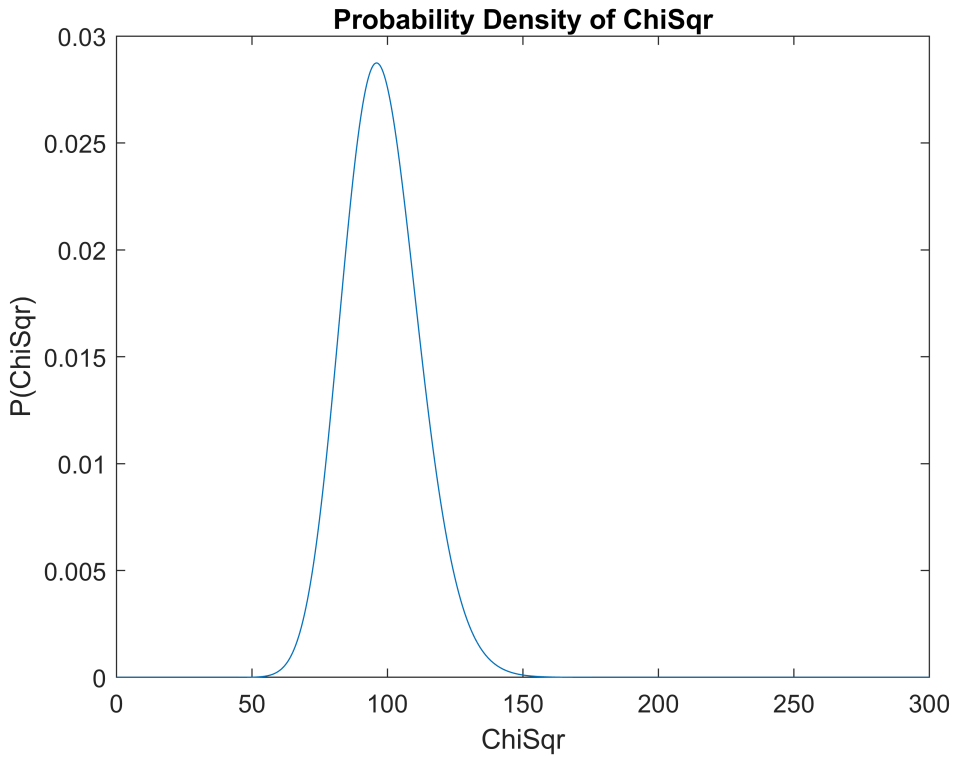


Plot 6: ln(counts) vs time with a Matlab fit plotted.

The χ^2 value of the fit was then calculated to be 194 and the $\frac{\chi^2}{\text{NDF}}$ was 1.98. These were calculated using a function defined at the end of this report (see Appendix 1).

This was compared to a prediction from the probability density of χ^2:

```
%calc the ChiSqr (non itterative) using function defined at end
[~, sumChiSqr2, DoF2, ChiSqr_NDF2] = chiSqr(t_short, ln_counts2_short, ...
    d_ln_counts2_short, fit2, 2);

%calculating the probability:
x = 0:0.2:300; %ChiSqr range
y = chi2pdf(x, DoF2); %probability density

figure(6) %plot of Probability density vs ChiSqr
plot(x, y);
xlabel('ChiSqr');
ylabel('P(ChiSqr)');
title('Probability Density of ChiSqr');
```



Plot 7: probability density of χ^2 for the same number of degrees of freedom as the data.

The plot seems to indicate that the data should have a χ^2 of ~100. This is much smaller than the value calculated. The probability was calculated to be $2.82 \cdot 10^{-8}$:

```
probability = 1 - chi2cdf(sumChiSqr2, DoF2); %extracting the probability
fprintf('The probability is %s.\n', num2str(probability))
```

```
The probability is 2.8195e-08.
```

Next, $T_{\frac{1}{2}}$ was calculated:

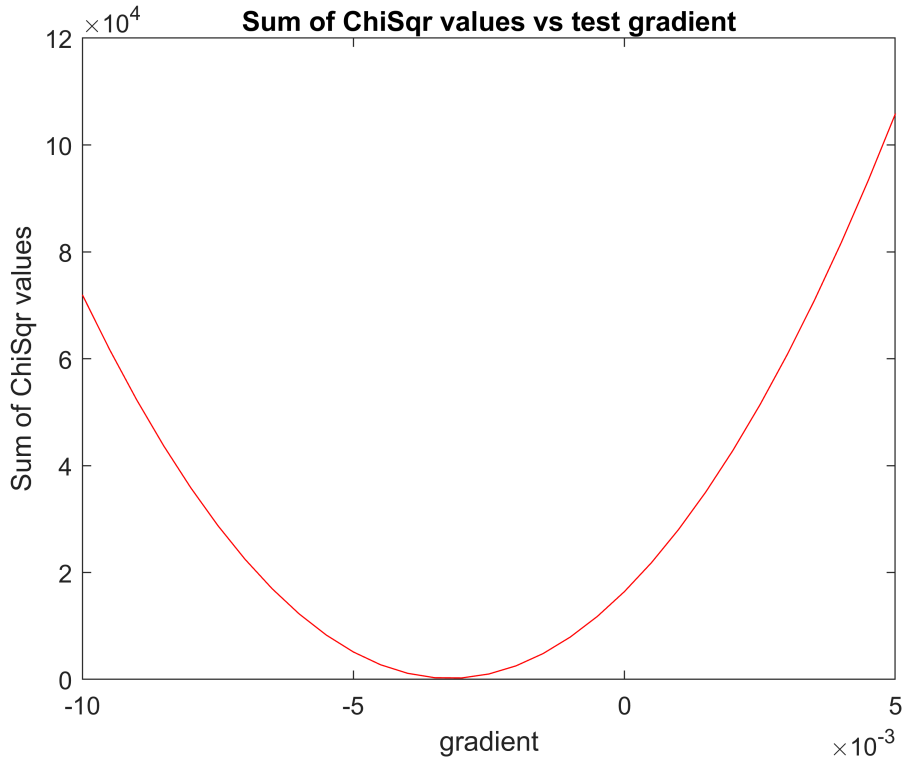
```
%Finding T_(1/2) using a non-iterative approach:
coefficients = coeffvalues(fit2); %coefficients of fit2
intervals = confint(fit2);
gradient1 = coefficients(1); %the gadient
intercept = coefficients(2); % the intercept
d_gradient1 = (intervals(2,1) - intervals(1,1))/2; %uncetainty on gradient
lambda1 = -gradient1; %lambda = -(gradient)
T1 = log(2) / lambda1; %half life 1 in s
d_T1 = d_gradient1 * T1 / lambda1; %uncertainty in half life 1
T1_min = T1 /60; %half life 1 in min
d_T1_min = d_T1 / 60; %uncetainty in half life 1 in min
```

The calculated λ was $0.0033 \pm 0.0002\ s^{-1}$ and $T_{\frac{1}{2}}$ was $210 \pm 10\,s$ or $3.5 \pm 0.2\,\text{min}$.

3b: Iterative approach to calculating $T_{\frac{1}{2}}$

An array of gradients was created and each iteration was used to calculate a test (counts) data set. The χ^2 value was calculated for each iteration and added to a column vector. This was then plotted, and the minimum extracted. This value was the gradient with the minimum (optimal) χ^2. This was at $\text{gradient} = -0.00322$ and $\chi^2 = 188$. The gradient was then used to calculate λ and $T_{\frac{1}{2}}$ using the same method as for part 3c. These were $\lambda = 0.00322\,s^{-1}$ and $T_{\frac{1}{2}} = 215\,s \text{ or } 3.58\,\text{min}$. These results are summaried in *Table 1* at the end of this section.

```
%looping over an arrary of test gradients:
loop_index = 0;
for test_gradient = -0.01:0.0005:0.005 %the array of test gradients to be iterated over.
    % start:stepSize:finish
    loop_index = loop_index + 1;
    sumChiSqr3(loop_index,1) = sum(((ln_counts2_short - (test_gradient * t_short + intercept))
end

figure(7)
x2(:,1) = -0.01:0.0005:0.005; %the gradients iterated over
plot(x2, sumChiSqr3, '-r')
xlabel('gradient')
ylabel('Sum of ChiSqr values')
title('Sum of ChiSqr values vs test gradient')
```



Plot 8: Sum of χ^2 for the data vs the gradient being tested. This was a minimum at gradient = -0.0032.

The minimum of the plot was found:

```
[fit3, ~] = fit(x2, sumChiSqr3, 'poly2'); %fitting the curve so parameters can be used
coefficients3 = coeffvalues(fit3); %extracting the coefficients
p1 = coefficients3(1);
p2 = coefficients3(2);
p3 = coefficients3(3);

%The best ChiSqr/NDF value is at the minimum. This is the gradient wanted
%Differentiating to find the minimum:
syms x real; %creating symbolic variable x, assigning it real
f = p1*x^2 + p2*x +p3;
g = diff(f, x);
g0 = solve(g,x);
min = double(g0); %this is the minimum, ie the gradient which produces the best ChiSqr/NDF

%the ChiSqr of this fit:
sumChiSqr3 = sum(((ln_counts2_short - (t_short*min + intercept))./d_ln_counts2_short).^2);
ChiSqr_NDF3 = sumChiSqr3 / (n2 - 2);

%calculating lambda and the half life
lambda2 = -min; %lambda = -(optimal gradient)
T2 = log(2) / lambda2; %half life 1 in s
%d_T1 = -d_gradient1 * T1 / lambda1; %uncertainty in half life 1
T2_min = T2 /60; %half life 1 in min
%dT_1_min = d_T1 / 60; %uncetainty in half life 1 in min
```

```
%table of results:
fitType = {'Matlab fitting'; 'Iterative approach'};
lambda = [lambda1; lambda2];
d_lambda = [d_gradient1; 0];
half_life_s = [T1; T2];
error1 = [d_T1; 0];
half_life_mins = [T1_min; T2_min];
error2 = [d_T1_min; 0];
ChiSqr = [sumChiSqr2; sumChiSqr3];
ChiSqr_by_NDF = [ChiSqr_NDF2; ChiSqr_NDF3];

results = table(fitType, lambda, d_lambda, half_life_s, error1, half_life_mins, ...
    error2, ChiSqr, ChiSqr_by_NDF);
disp(results)
```

fitType	lambda	d_lambda	half_life_s	error1	half_life_mins	error2	ChiSqr
'Matlab fitting'	0.0032857	0.00019983	210.96	12.83	3.516	0.21384	193.85
'Iterative approach'	0.0032233	0	215.04	0	3.5841	0	187.76

Table 2: Table of results for both the non-iterative and iterative fitting techniques.

Appendix 1

This function was used to evaluate the fits and to obtain values for further calculations in Question 1 and Question 3.

```
function [ChiSqr, sumChiSqr, DoF, ChiSqr_NDF] = chiSqr(x, y, d_y, fitName, no_parameters)
y_fit = fitName(x);
n = length(x);
ChiSqr = ((y - y_fit) ./ d_y ).^2;
sumChiSqr = sum(ChiSqr);
DoF = n - no_parameters;
ChiSqr_NDF = sumChiSqr / DoF;
end
```