HPGe Detector Appendix

Alessandro Maraio & Samuel Morgan

The following document is our appendix to our lab books that contain all of the figures that are being produced by our various codes, and the codes themselves, that have been produced over the four weeks of the experiment. Here, detailed plots are produced for all of the individual fits that have been done on all of the peaks, with all of the various fitting functions. Conclusions can be drawn from these figures, and so they are included here.

A1 Fitted Peaks

For all of the known sources, we had to fit all of the peaks with various fitting functions to extract the central channel number for that peak. From this, it can then be fed into our tools that calibrates the detector properly, as we can produce a further fit that is of channel numbers as a function of peak energy.

To properly test the fits, we had to fit the data with three different functions to assess the quality of the fits, as well as how the mean and standard deviations of the fit varies with the fitting function. Then for each fit, we first ran a 'rough' fit, that is where we have manually gone through the plots, and isolated each peak extracting the approximate lower-bound, mean, and upper-bound of each peak. This then gave us a rough estimation of the location & mean of each peak, which was then passed into the fitting tool. From there, the new mean and standard deviation was extracted. From these new values, we then produced a 'zoomed' fit, by limiting the data to $x = \mu \pm 2\sigma$ for each peak, which then constrained the peak correctly, producing a more accurate fit.
The reason behind this double-fitting process is that for some peaks, there are significant noise and background effects that causes the count numbers to behave significantly non-Gaussian, which can affect the location of the mean for the peak. By containing the fit to only the central maxima, we can reduce these effects significantly – and so the mean that we extract from the fit is much more likely to be that for the actual data, rather than any effects introducing any shifts.

All of the following plots are for the 24 hour data obtained, which is reflected by the high count numbers. By running each source for longer, we should be able to reduce the effects of any statistical noise fluctuations that may induce a slight shift in the data.

The following plots are broken down first by source, in decreasing atomic number, and then further broken down into subsections by the fit type used to produce each plot. For each fit, there are the the fit statistics and the mean & standard deviation for the peak.

A1.1 Barium-133

The ^{133}Ba source had the most amount of identifiable peaks, totalling seven. This gave us a large number of peaks to fit, and so improved the accuracy of our energy calibration. All of these peaks were considered low energy, as they all occur below 400 keV. This is shown in the figure below

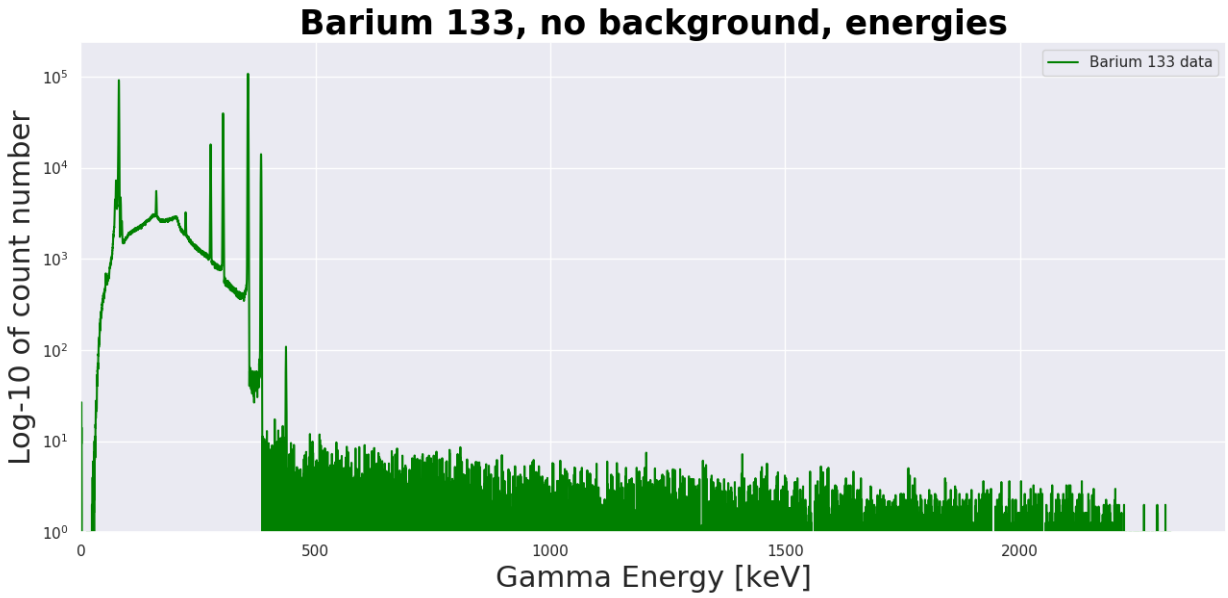
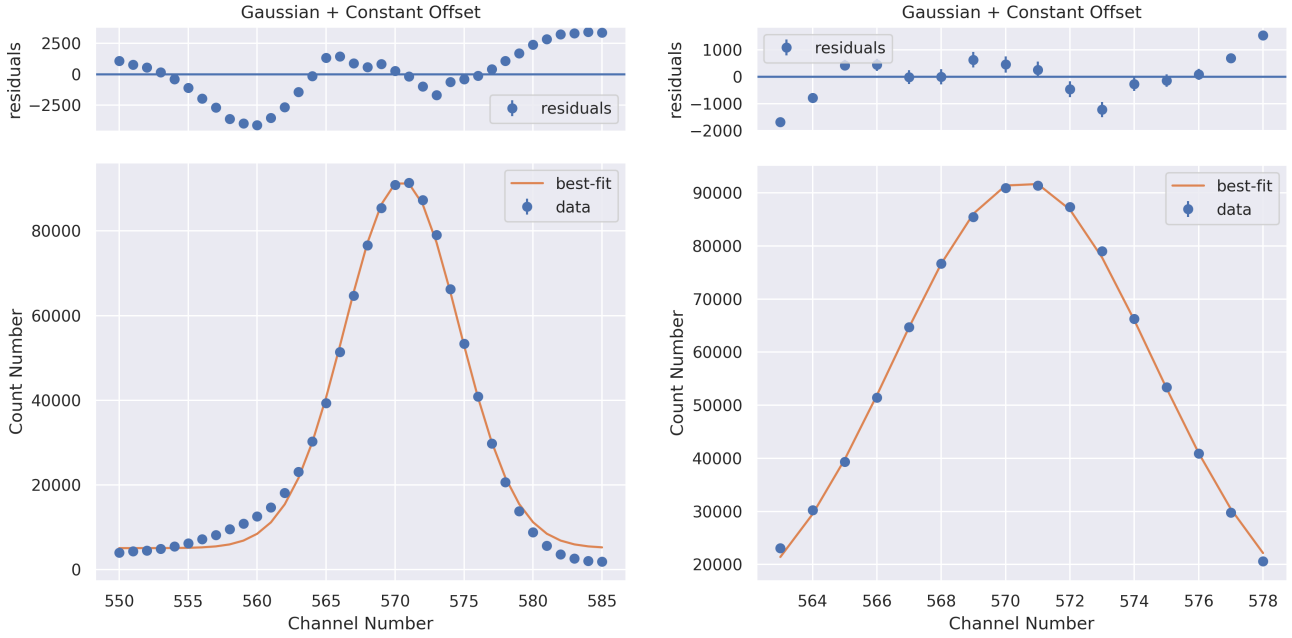


Figure A.1: The final spectrum obtained for the ^{133}Ba source run for 24 hours. This has been converted to energies by using the quadratic energy calibration.

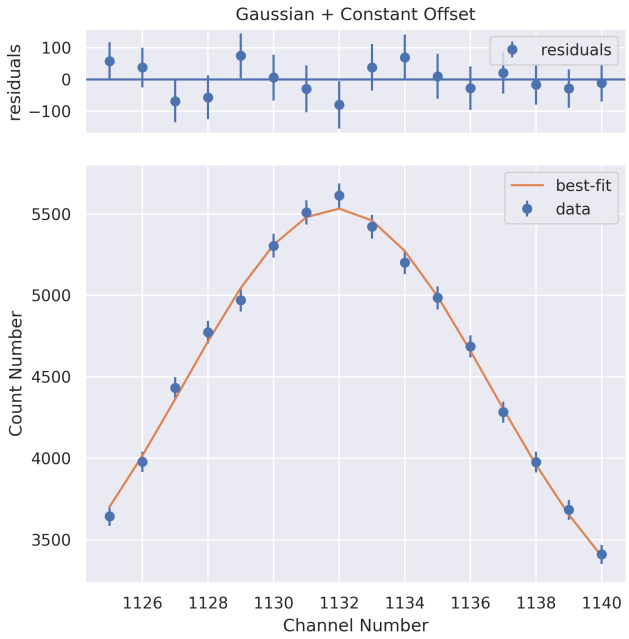
A1.1.1 Gaussian + Offset Fit



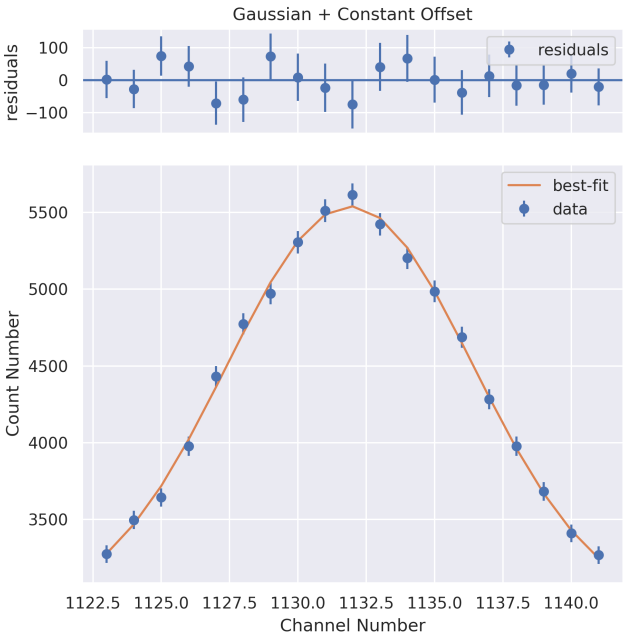
(a) Full peak with fit. $\chi^2 = 29044$, $\chi^2_\nu = 854$, Prob = 0%, $\mu = 570.5$, $\sigma = 4.12$

(b) Zoomed in peak with fit. $\chi^2 = 310.6$, $\chi^2_\nu = 22.2$, Prob = 0%, $\mu = 570.6$, $\sigma = 4.00$

Figure A.2: Fit of full & zoomed in peak of ^{133}Ba 81 keV peak

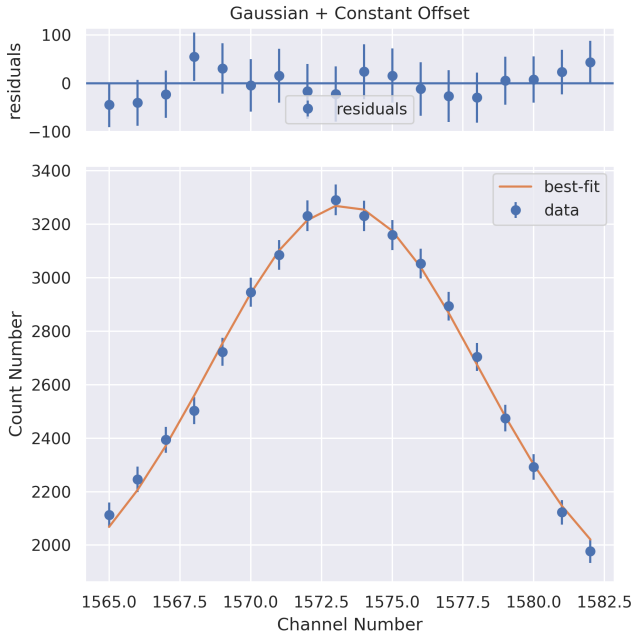


(a) Full peak with fit. $\chi^2 = 7.26$, $\chi^2_\nu = 0.52$, Prob = 92.4%, $\mu = 1131.9$, $\sigma = 4.7$

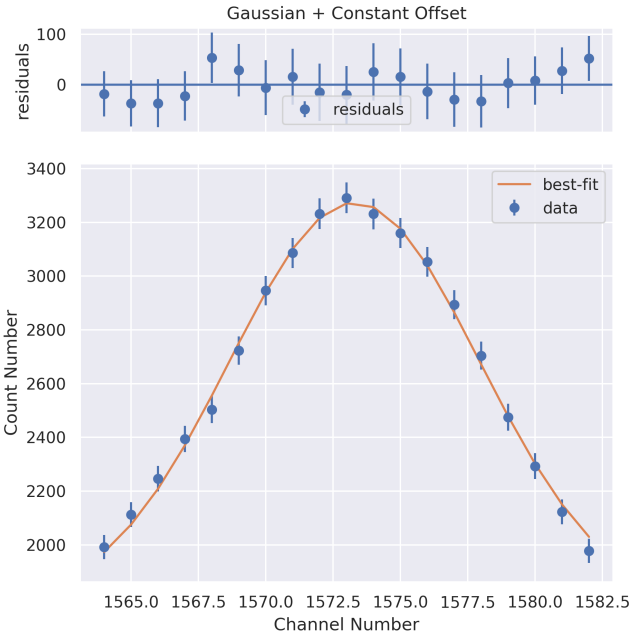


(b) Zoomed in peak with fit. $\chi^2 = 8.16$, $\chi^2_\nu = 0.48$, Prob = 96.3%, $\mu = 1131.9$, $\sigma = 4.51$

Figure A.3: Fit of full & zoomed in peak of ^{133}Ba 161 keV peak

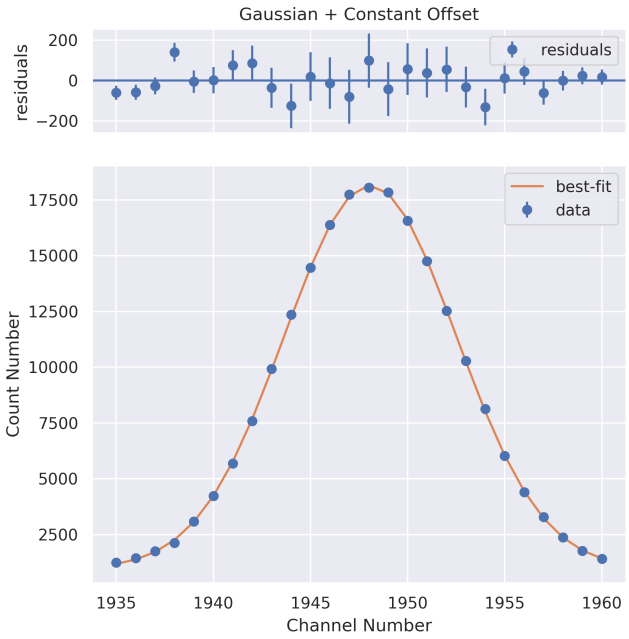


(a) Full peak with fit. $\chi^2 = 5.88$, $\chi^2_\nu = 0.36$, Prob = 98.93%, $\mu = 1573.29$, $\sigma = 4.73$

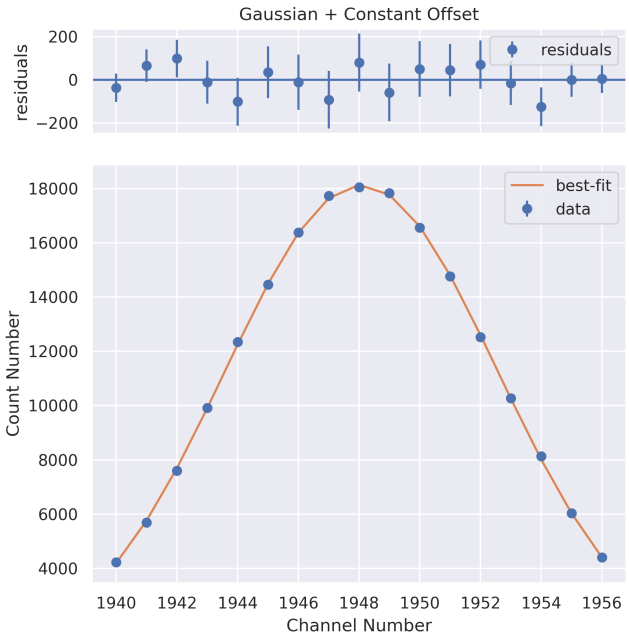


(b) Zoomed in peak with fit. $\chi^2 = 6.18$, $\chi^2_\nu = 0.36$, Prob = 99.19%, $\mu = 1573.23$, $\sigma = 4.65$

Figure A.4: Fit of full & zoomed in peak of ^{133}Ba 223 keV peak

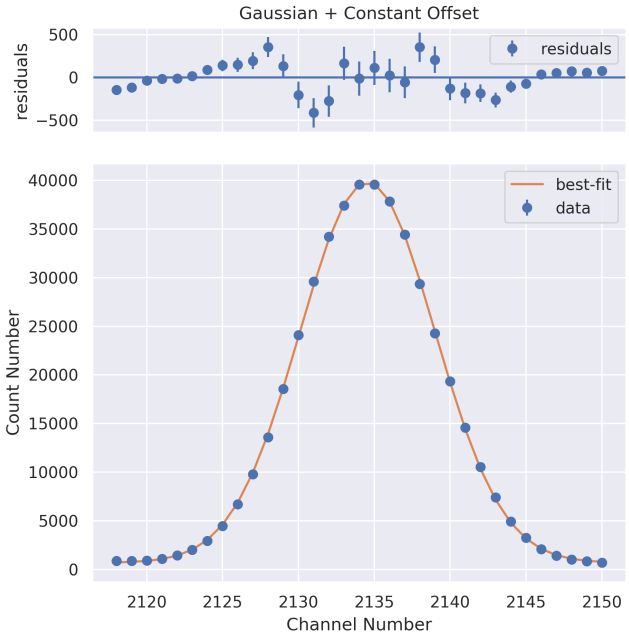


(a) Full peak with fit. $\chi^2 = 24.2$, $\chi^2_\nu = 1.01$, Prob = 44.9%, $\mu = 1948.1$, $\sigma = 4.43$

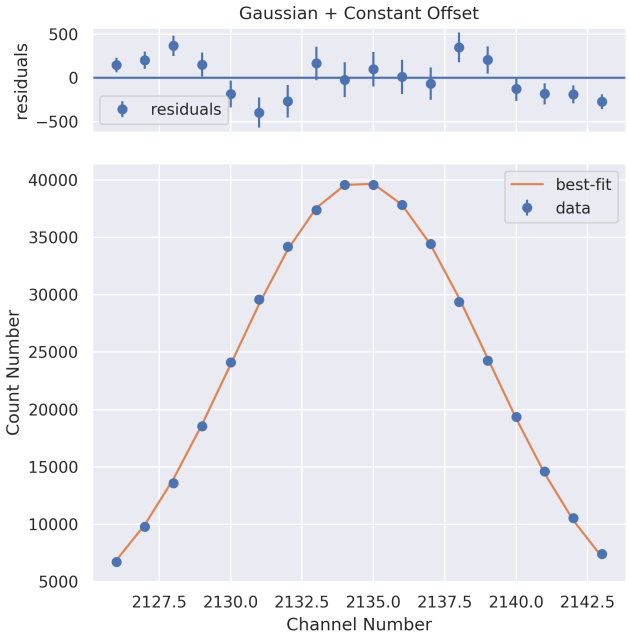


(b) Zoomed in peak with fit. $\chi^2 = 6.92$, $\chi^2_\nu = 0.46$, Prob = 96.0%, $\mu = 1948.1$, $\sigma = 4.48$

Figure A.5: Fit of full & zoomed in peak of ^{133}Ba 276 keV peak

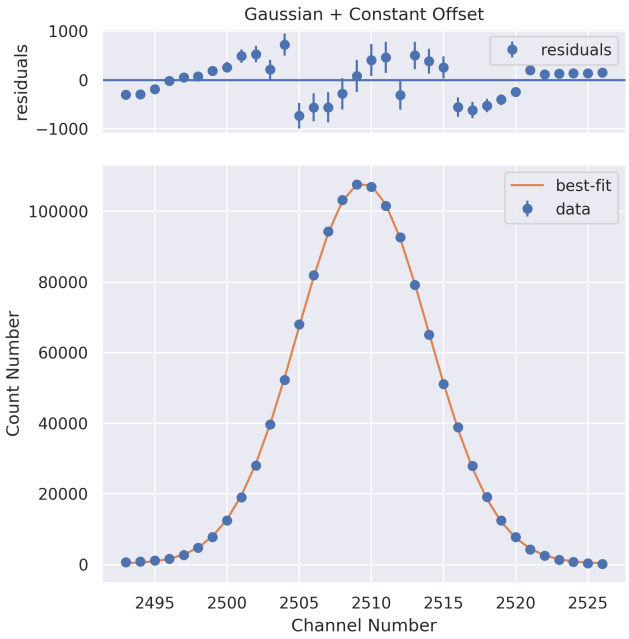


(a) Full peak with fit. $\chi^2 = 121$, $\chi^2_\nu = 4$, Prob = 0%, $\mu = 2134.55$, $\sigma = 4.45$

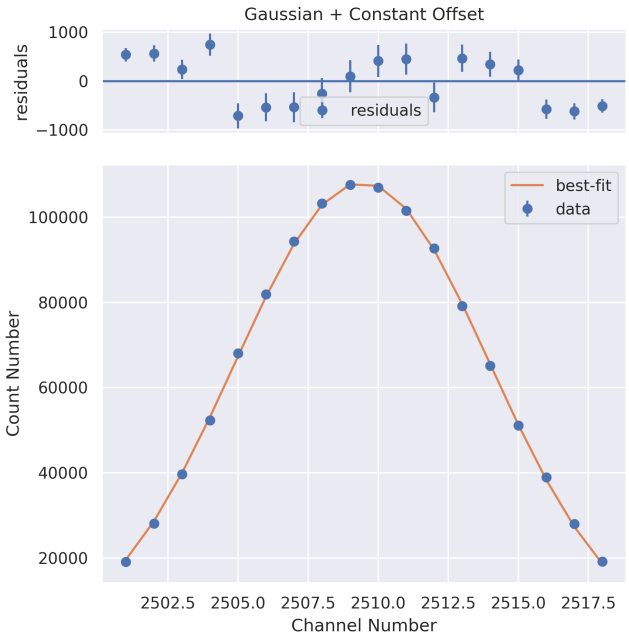


(b) Zoomed in peak with fit. $\chi^2 = 51.0$, $\chi^2_\nu = 3.2$, Prob = 0.02%, $\mu = 2134.55$, $\sigma = 4.56$

Figure A.6: Fit of full & zoomed in peak of ^{133}Ba 303 keV peak

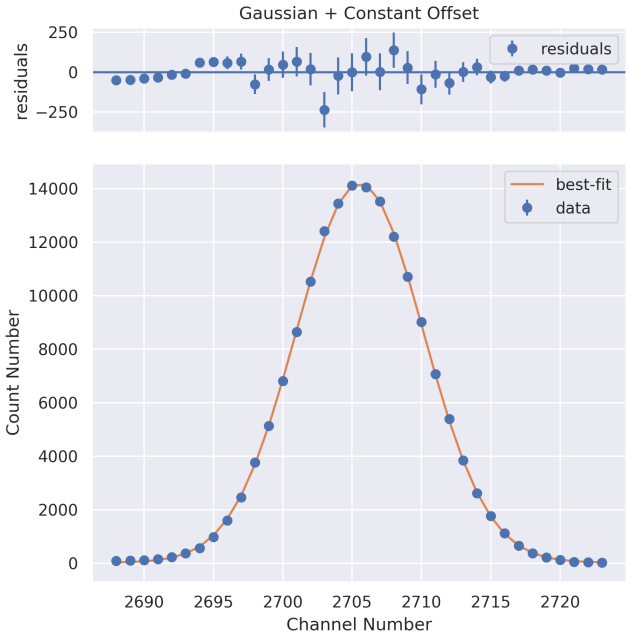


(a) Full peak with fit. $\chi^2 = 608$, $\chi^2_\nu = 19$, Prob = 0%, $\mu = 2509.4$, $\sigma = 4.5$

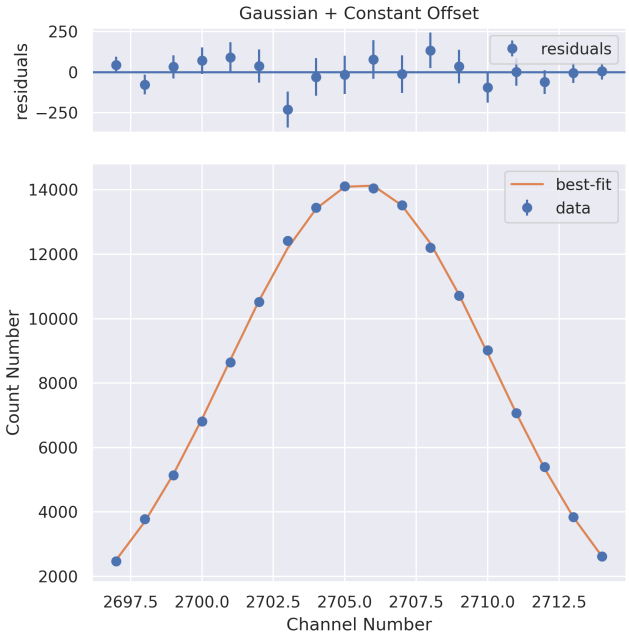


(b) Zoomed in peak with fit. $\chi^2 = 100.1$, $\chi^2_\nu = 6.3$, Prob = 0%, $\mu = 2509.4$, $\sigma = 4.5$

Figure A.7: Fit of full & zoomed in peak of ^{133}Ba 356 keV peak



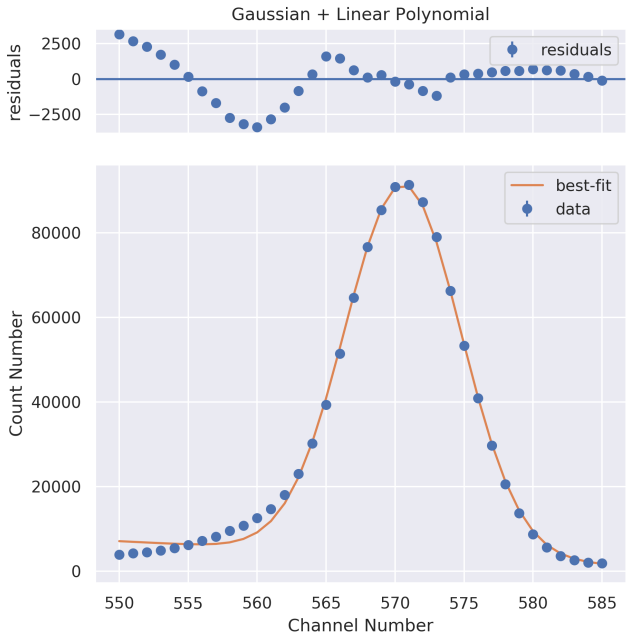
(a) Full peak with fit. $\chi^2 = 127$, $\chi^2_\nu = 4$, Prob = 0%, $\mu = 2705.56$, $\sigma = 4.59$



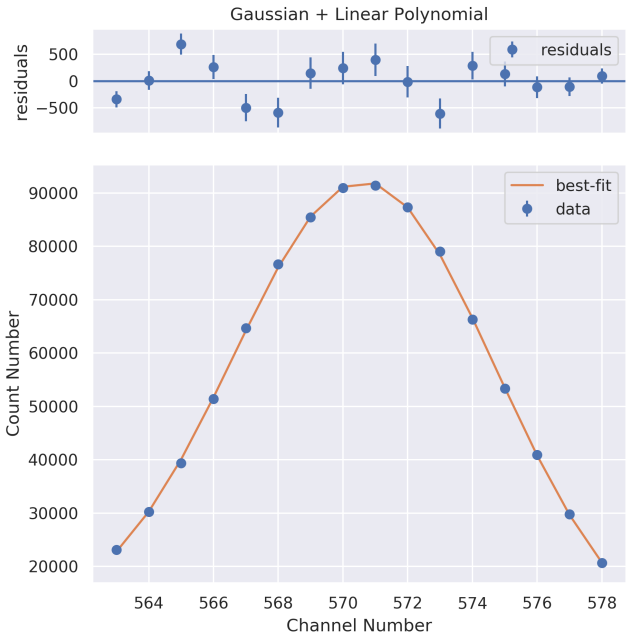
(b) Zoomed in peak with fit. $\chi^2 = 12.5$, $\chi^2_\nu = 0.8$, Prob = 70.9%, $\mu = 2705.55$, $\sigma = 4.64$

Figure A.8: Fit of full & zoomed in peak of ^{133}Ba 384 keV peak

A1.1.2 Gaussian + Linear Fit

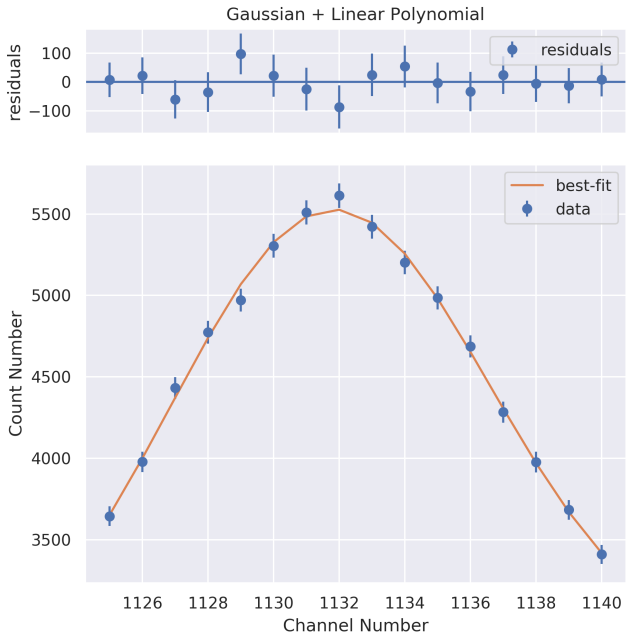


(a) Full peak with fit. $\chi^2 = 10435$, $\chi^2_\nu = 307$, Prob = 0%, $\mu = 570.59$, $\sigma = 4.20$

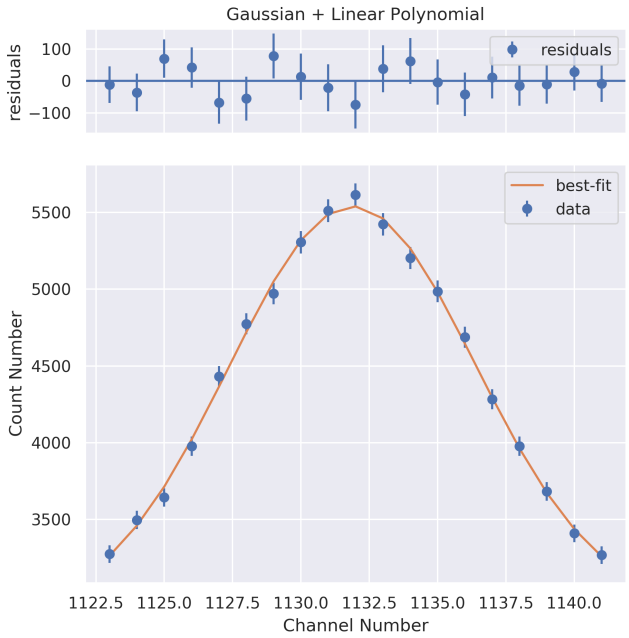


(b) Zoomed in peak with fit. $\chi^2 = 36.6$, $\chi^2_\nu = 2.6$, Prob = 0.08%, $\mu = 570.68$, $\sigma = 4.05$

Figure A.9: Fit of full & zoomed in peak of ^{133}Ba 81 keV peak

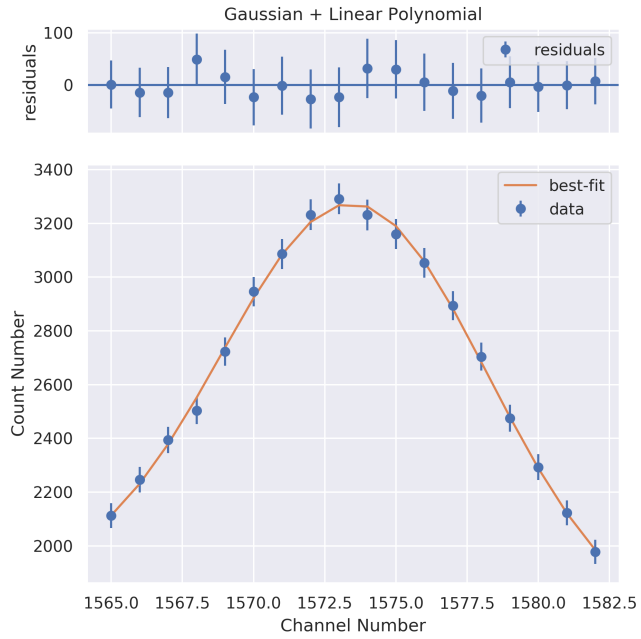


(a) Full peak with fit. $\chi^2 = 5.78$, $\chi^2_\nu = 0.41$, Prob = 97.18%, $\mu = 1131.73$, $\sigma = 4.85$

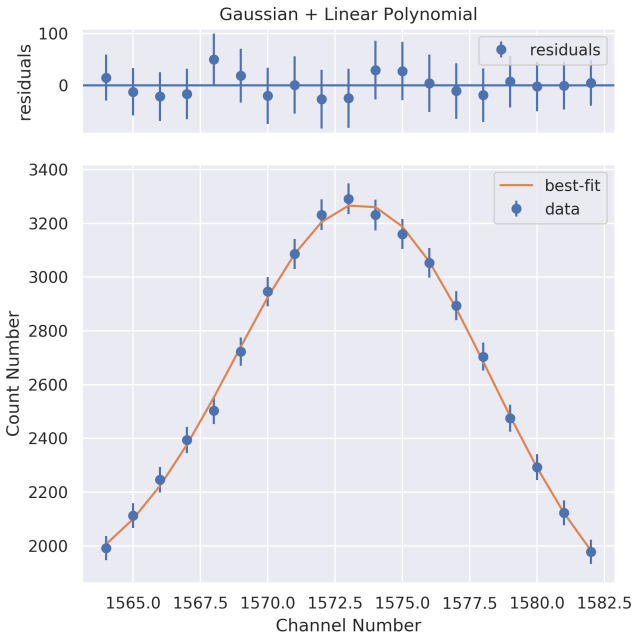


(b) Zoomed in peak with fit. $\chi^2 = 8.02$, $\chi^2_\nu = 0.47$, Prob = 96.61%, $\mu = 1131.87$, $\sigma = 4.51$

Figure A.10: Fit of full & zoomed in peak of ^{133}Ba 161 keV peak

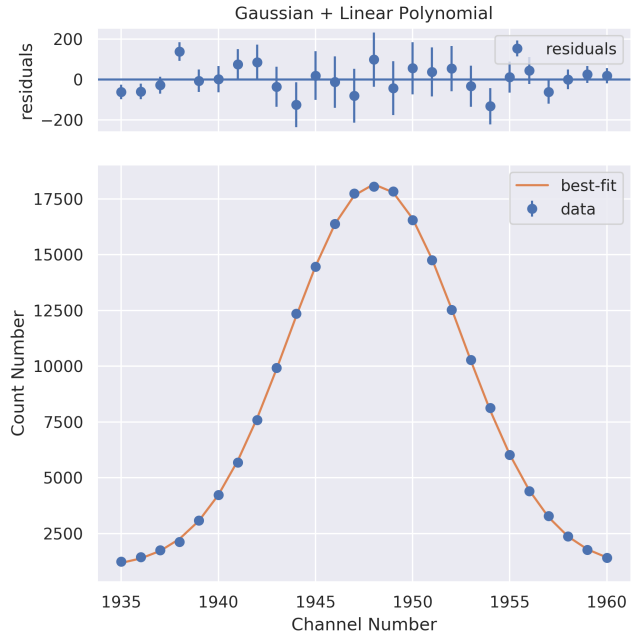


(a) Full peak with fit. $\chi^2 = 2.65$, $\chi^2_\nu = 0.17$, Prob = 99.99%, $\mu = 1573.55$, $\sigma = 4.66$

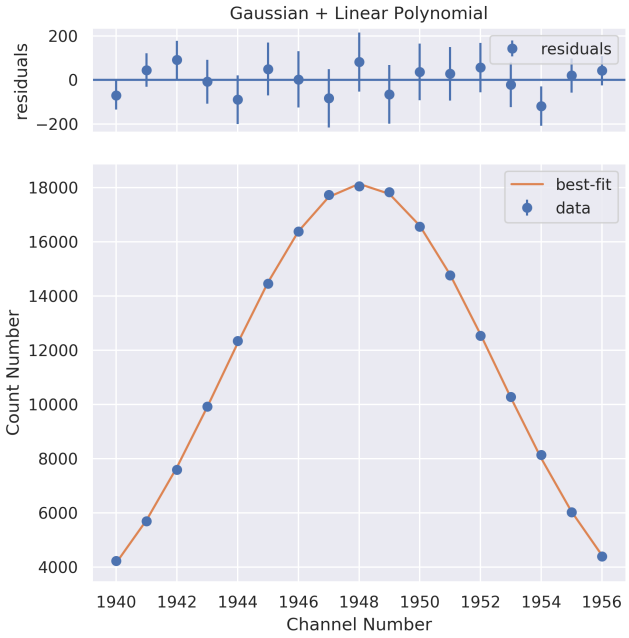


(b) Zoomed in peak with fit. $\chi^2 = 2.93$, $\chi^2_\nu = 0.17$, Prob = 99.99%, $\mu = 1573.52$, $\sigma = 4.74$

Figure A.11: Fit of full & zoomed in peak of ^{133}Ba 223 keV peak

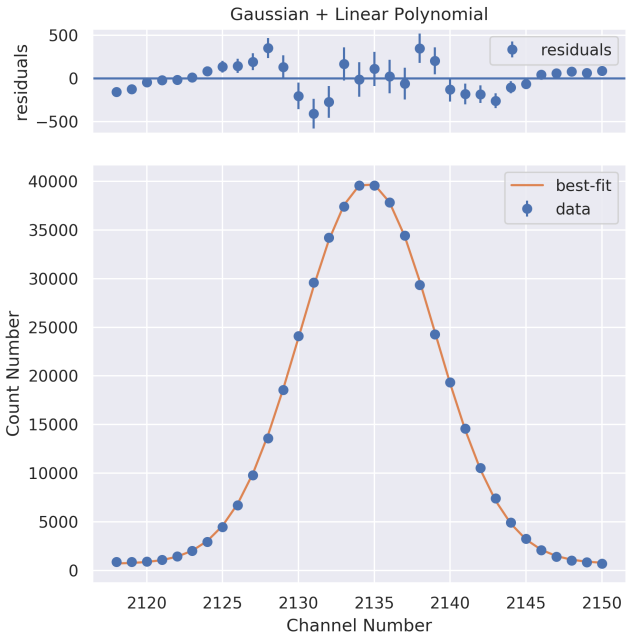


(a) Full peak with fit. $\chi^2 = 24.39$, $\chi^2_\nu = 1.02$, Prob = 43.93%, $\mu = 1948.08$, $\sigma = 4.43$

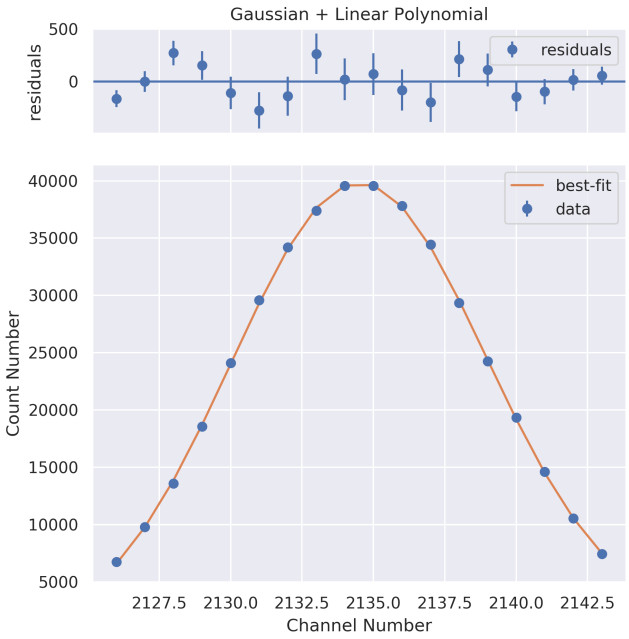


(b) Zoomed in peak with fit. $\chi^2 = 7.08$, $\chi^2_\nu = 0.47$, Prob = 95.54%, $\mu = 1948.06$, $\sigma = 4.47$

Figure A.12: Fit of full & zoomed in peak of ^{133}Ba 276 keV peak

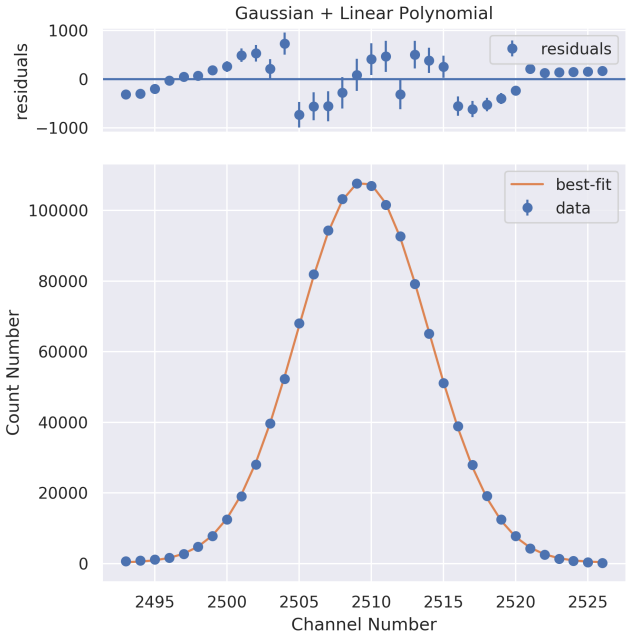


(a) Full peak with fit. $\chi^2 = 131.45$, $\chi^2_\nu = 4.24$, Prob = 0%, $\mu = 2134.55$, $\sigma = 4.45$

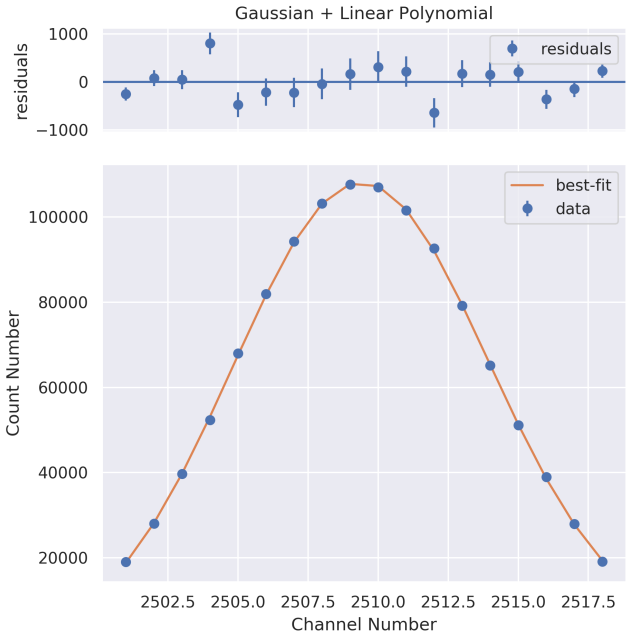


(b) Zoomed in peak with fit. $\chi^2 = 21.84$, $\chi^2_\nu = 1.36$, Prob = 14.85%, $\mu = 2134.49$, $\sigma = 4.45$

Figure A.13: Fit of full & zoomed in peak of ^{133}Ba 303 keV peak

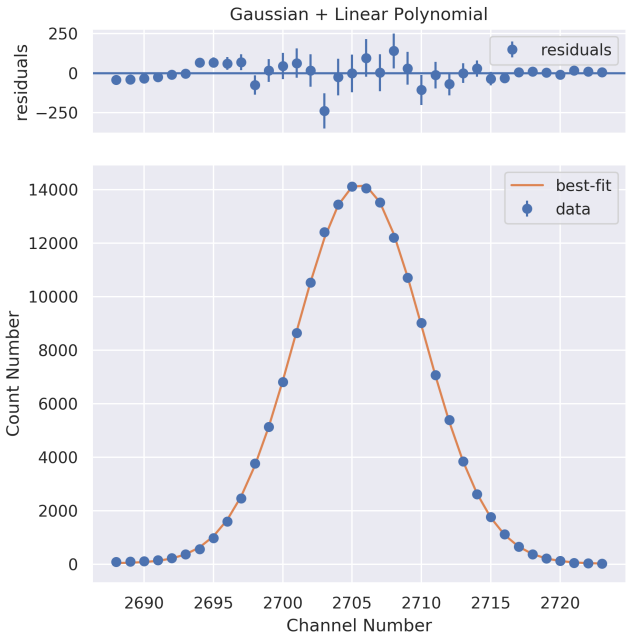


(a) Full peak with fit. $\chi^2 = 658.02$, $\chi^2_\nu = 20.56$, Prob = 0%, $\mu = 2509.44$, $\sigma = 4.55$

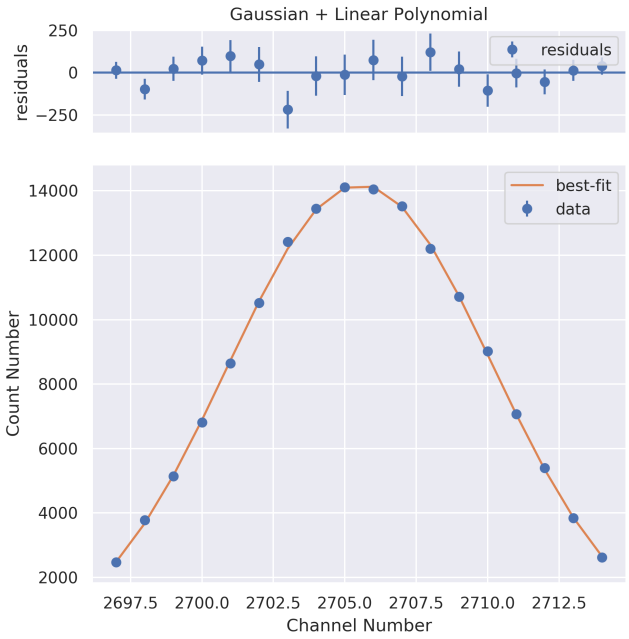


(b) Zoomed in peak with fit. $\chi^2 = 34.92$, $\chi^2_\nu = 2.18$, Prob = 0.41%, $\mu = 2509.38$, $\sigma = 4.55$

Figure A.14: Fit of full & zoomed in peak of ^{133}Ba 356 keV peak



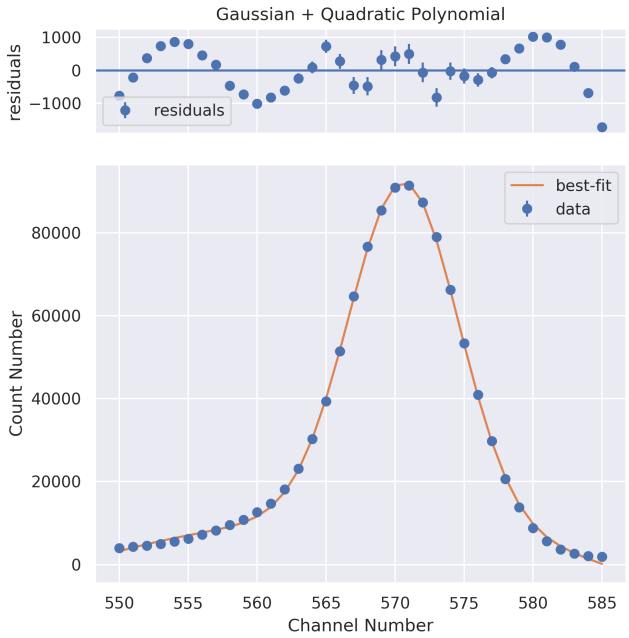
(a) Full peak with fit. $\chi^2 = 85.03$, $\chi^2_\nu = 2.50$, Prob = 0%, $\mu = 2705.06$, $\sigma = 4.59$



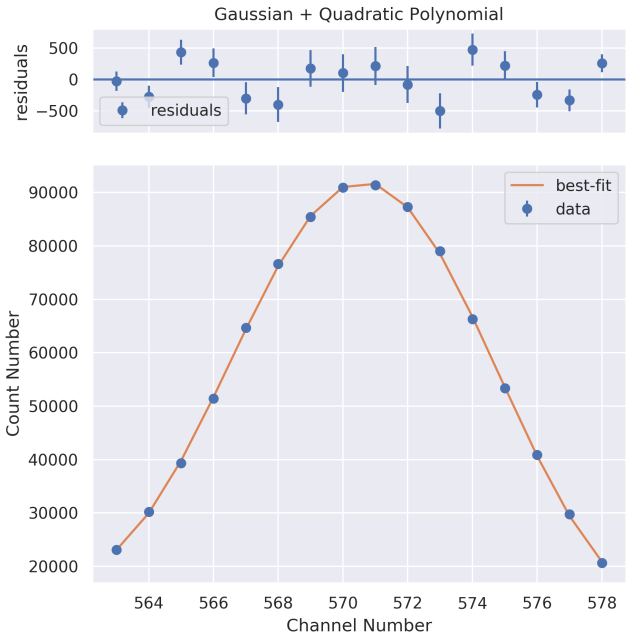
(b) Zoomed in peak with fit. $\chi^2 = 12.68$, $\chi^2_\nu = 0.79$, Prob = 69.69%, $\mu = 2705.53$, $\sigma = 4.63$

Figure A.15: Fit of full & zoomed in peak of ^{133}Ba 384 keV peak

A1.1.3 Gaussian + Quadratic Fit

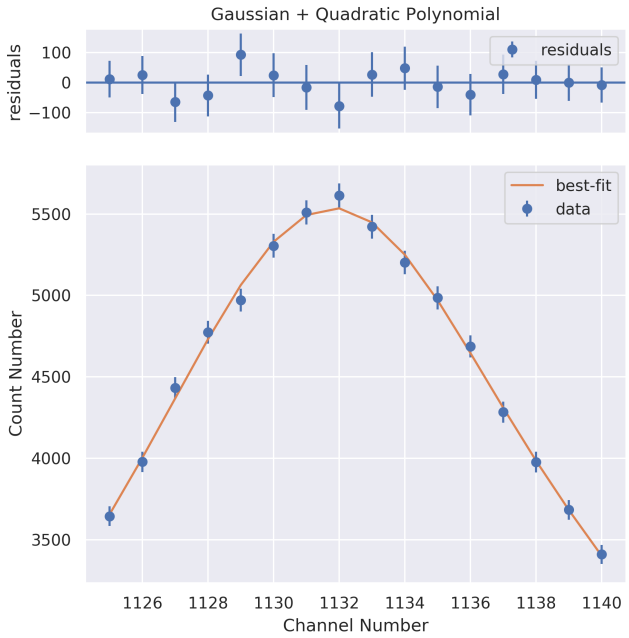


(a) Full peak with fit. $\chi^2 = 3108.72$, $\chi^2_\nu = 91.43$, Prob = 0%, $\mu = 570.65$, $\sigma = 3.96$

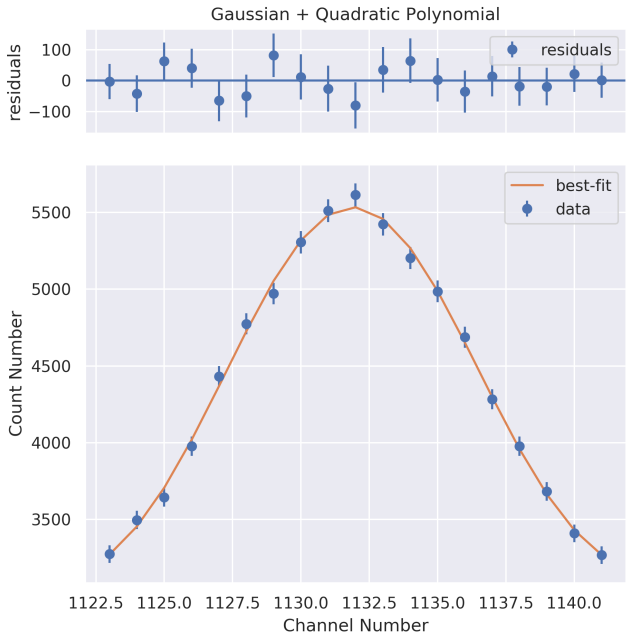


(b) Zoomed in peak with fit. $\chi^2 = 29.06$, $\chi^2_\nu = 2.08$, Prob = 1.03%, $\mu = 570.67$, $\sigma = 4.61$

Figure A.16: Fit of full & zoomed in peak of ^{133}Ba 81 keV peak

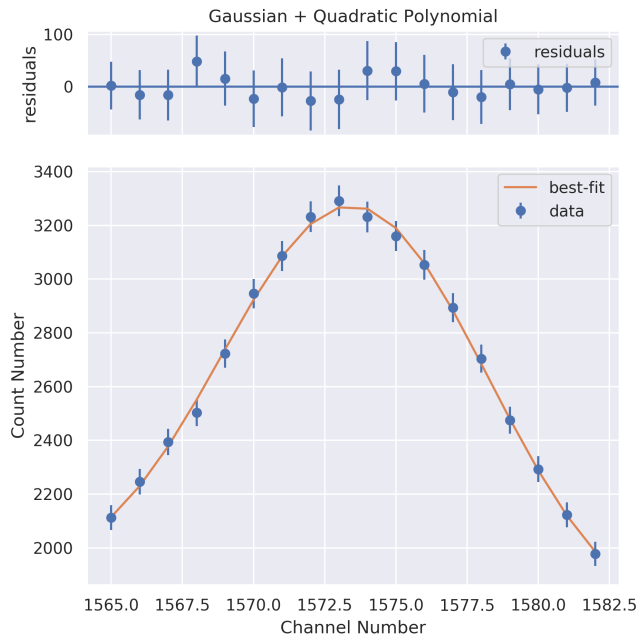


(a) Full peak with fit. $\chi^2 = 5.65$, $\chi^2_\nu = 0.40$, Prob = 97.45%, $\mu = 1131.70$, $\sigma = 3.89$

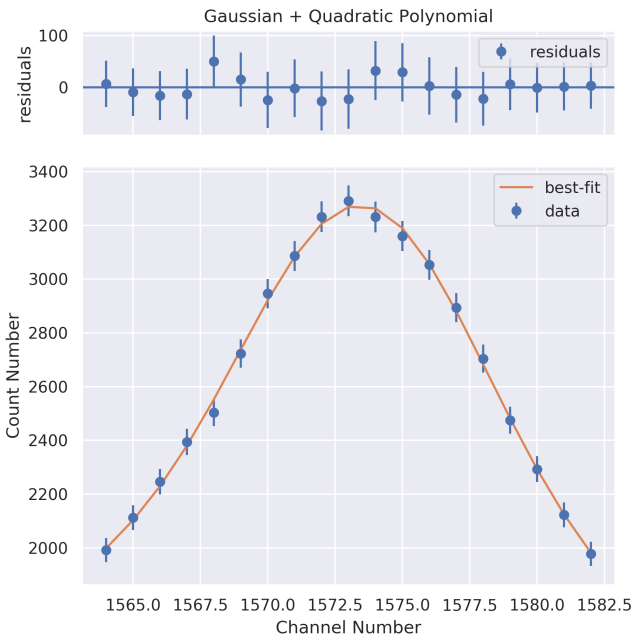


(b) Zoomed in peak with fit. $\chi^2 = 7.80$, $\chi^2_\nu = 0.46$, Prob = 97.07%, $\mu = 1131.88$, $\sigma = 4.69$

Figure A.17: Fit of full & zoomed in peak of ^{133}Ba 161 keV peak

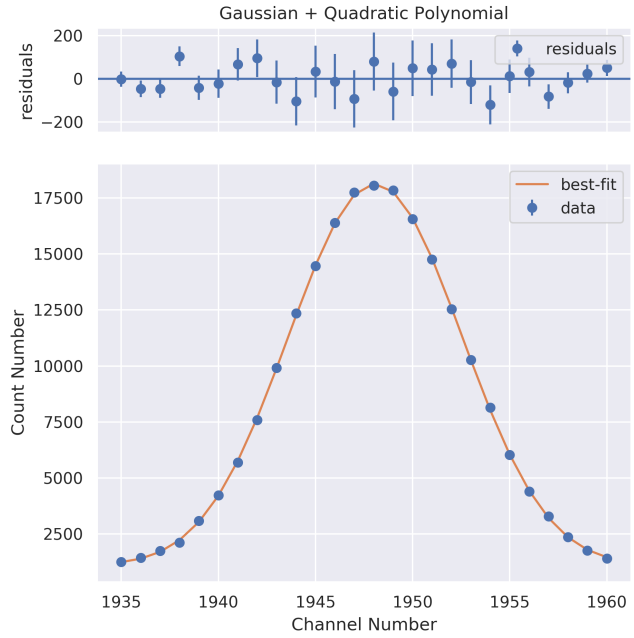


(a) Full peak with fit. $\chi^2 = 2.66$, $\chi^2_\nu = 0.17$, Prob = 99.99%, $\mu = 1573.55$, $\sigma = 4.82$

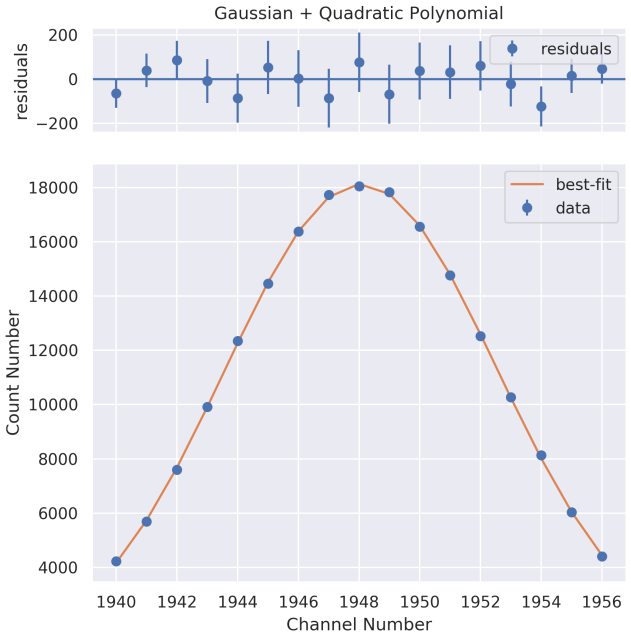


(b) Zoomed in peak with fit. $\chi^2 = 2.77$, $\chi^2_\nu = 0.16$, Prob = 100%, $\mu = 1573.53$, $\sigma = 4.34$

Figure A.18: Fit of full & zoomed in peak of ^{133}Ba 223 keV peak

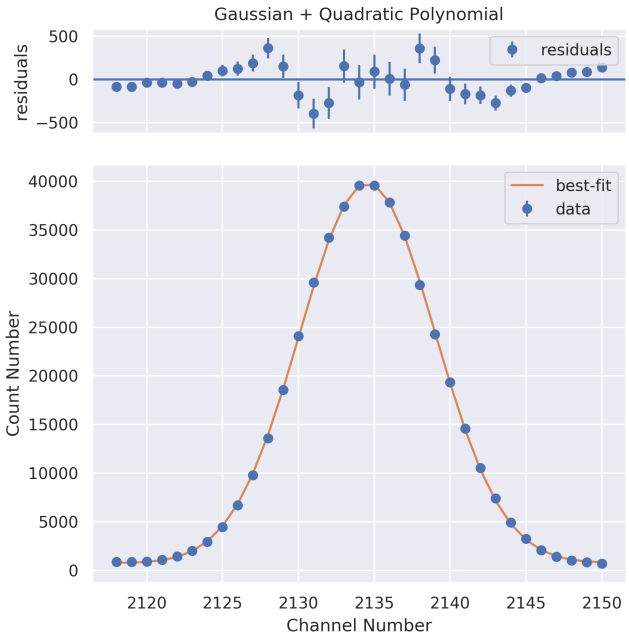


(a) Full peak with fit. $\chi^2 = 19.63$, $\chi^2_\nu = 0.82$, Prob = 71.76%, $\mu = 1948.07$, $\sigma = 4.49$

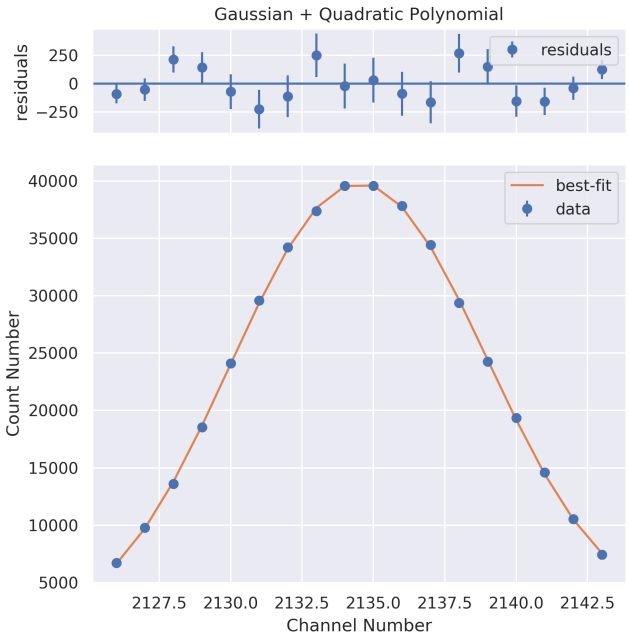


(b) Zoomed in peak with fit. $\chi^2 = 6.91$, $\chi^2_\nu = 0.46$, Prob = 96.00%, $\mu = 1948.06$, $\sigma = 4.54$

Figure A.19: Fit of full & zoomed in peak of ^{133}Ba 276 keV peak

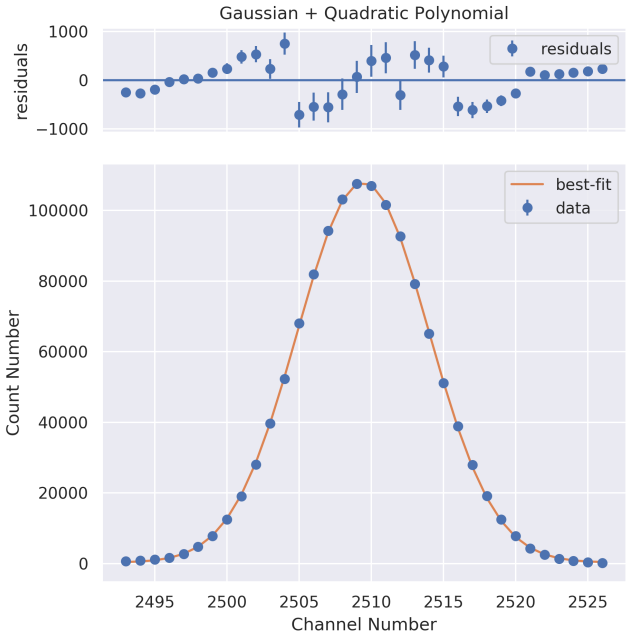


(a) Full peak with fit $\chi^2 = 123.06$, $\chi^2_\nu = 3.97$, Prob = 0.00%, $\mu = 2134.55$, $\sigma = 4.47$

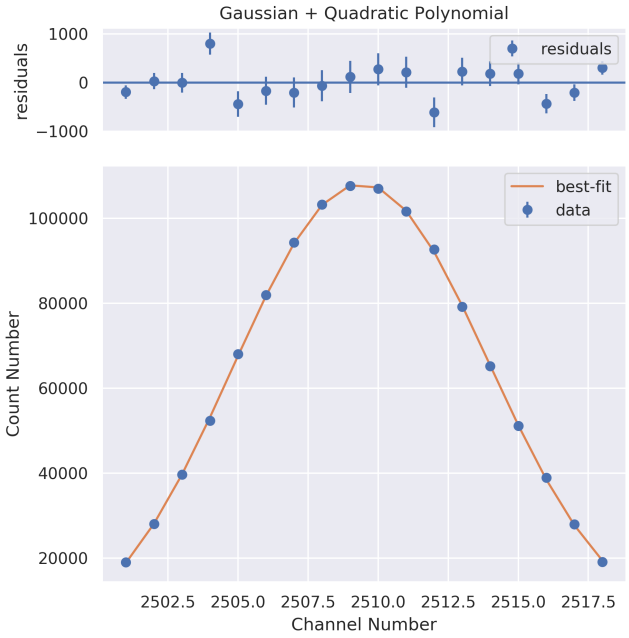


(b) Zoomed in peak with fit. $\chi^2 = 19.63$, $\chi^2_\nu = 1.23$, Prob = 23.74%, $\mu = 2134.49$, $\sigma = 4.73$

Figure A.20: Fit of full & zoomed in peak of ^{133}Ba 303 keV peak

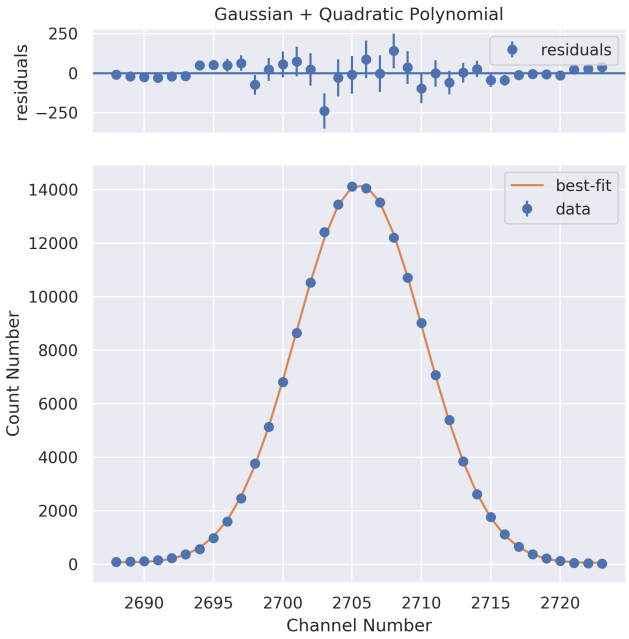


(a) Full peak with fit. $\chi^2 = 722.64$, $\chi^2_\nu = 22.58$, Prob = 0.00%, $\mu = 2509.44$, $\sigma = 4.56$

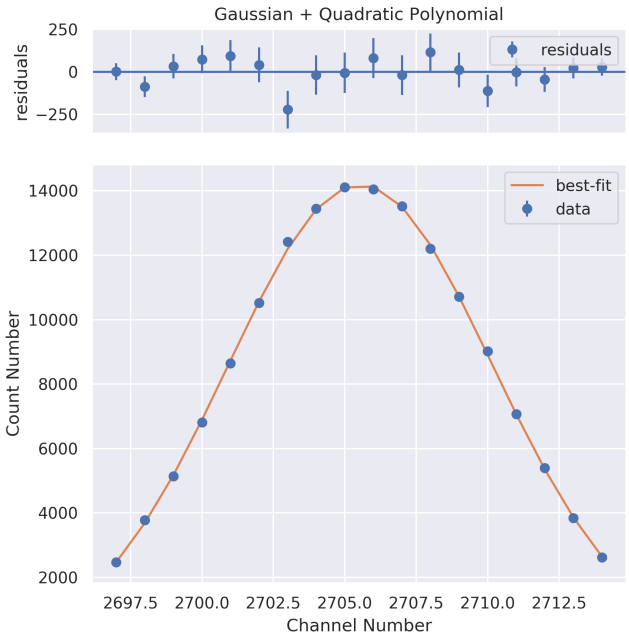


(b) Zoomed in peak with fit. $\chi^2 = 3.14$, $\chi^2_\nu = 2.26$, Prob = 0.28%, $\mu = 2509.38$, $\sigma = 4.65$

Figure A.21: Fit of full & zoomed in peak of ^{133}Ba 356 keV peak



(a) Full peak with fit. $\chi^2 = 119.72$, $\chi^2_\nu = 3.52$, Prob = 0.00%, $\mu = 2705.56$, $\sigma = 4.62$



(b) Zoomed in peak with fit. $\chi^2 = 12.07$, $\chi^2_\nu = 0.75$, Prob = 73.93%, $\mu = 2705.53$, $\sigma = 4.50$

Figure A.22: Fit of full & zoomed in peak of ^{133}Ba 384 keV peak

A1.2 Cobalt-60

The ^{60}Co source had two high-energy peaks which were very distinctive, and so quite easy to fit for all functions.

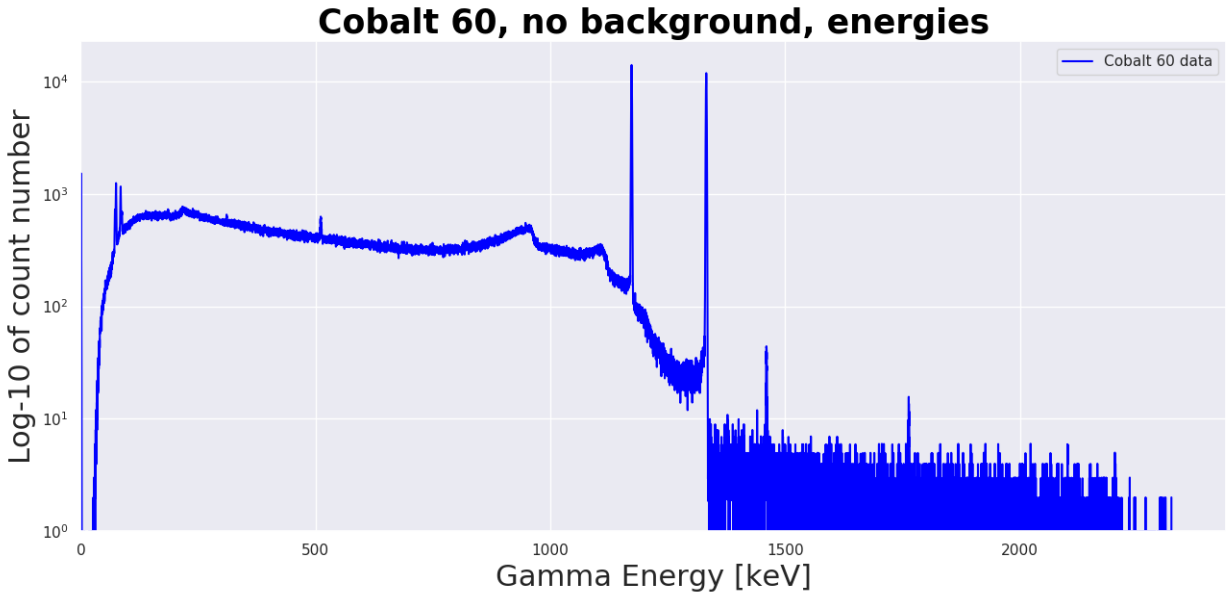
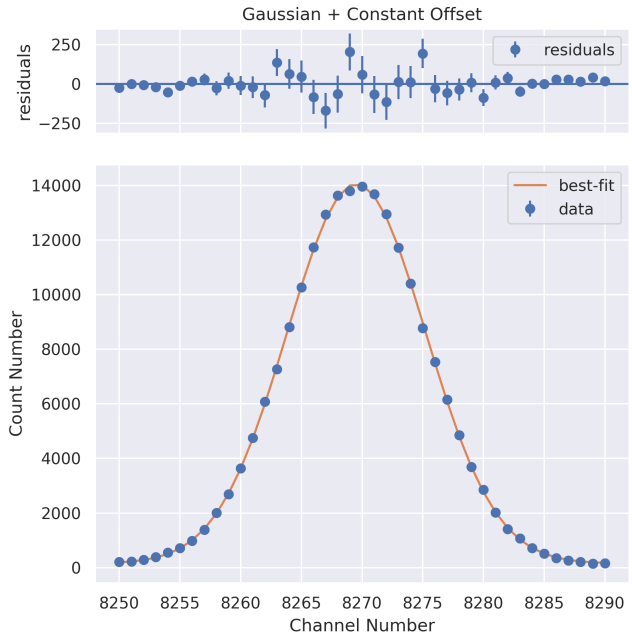
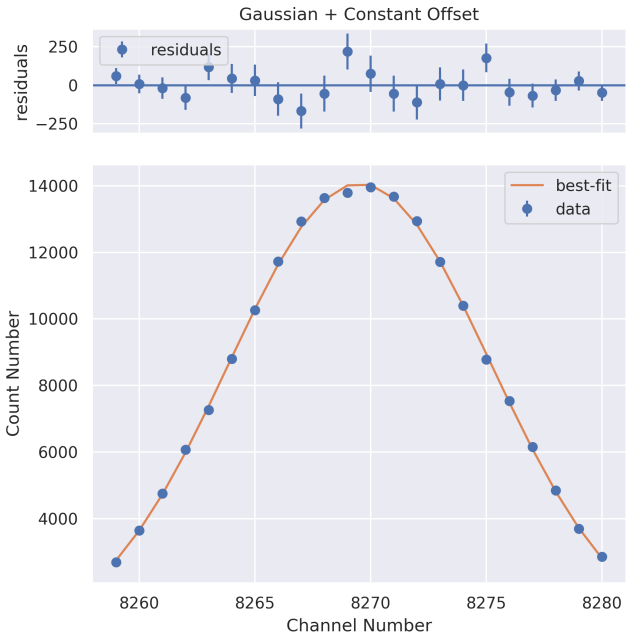


Figure A.23: The final spectrum obtained for the ^{60}Co source run for 24 hours. This has been converted to energies by using the quadratic energy calibration.

A1.2.1 Gaussian + Offset Fit

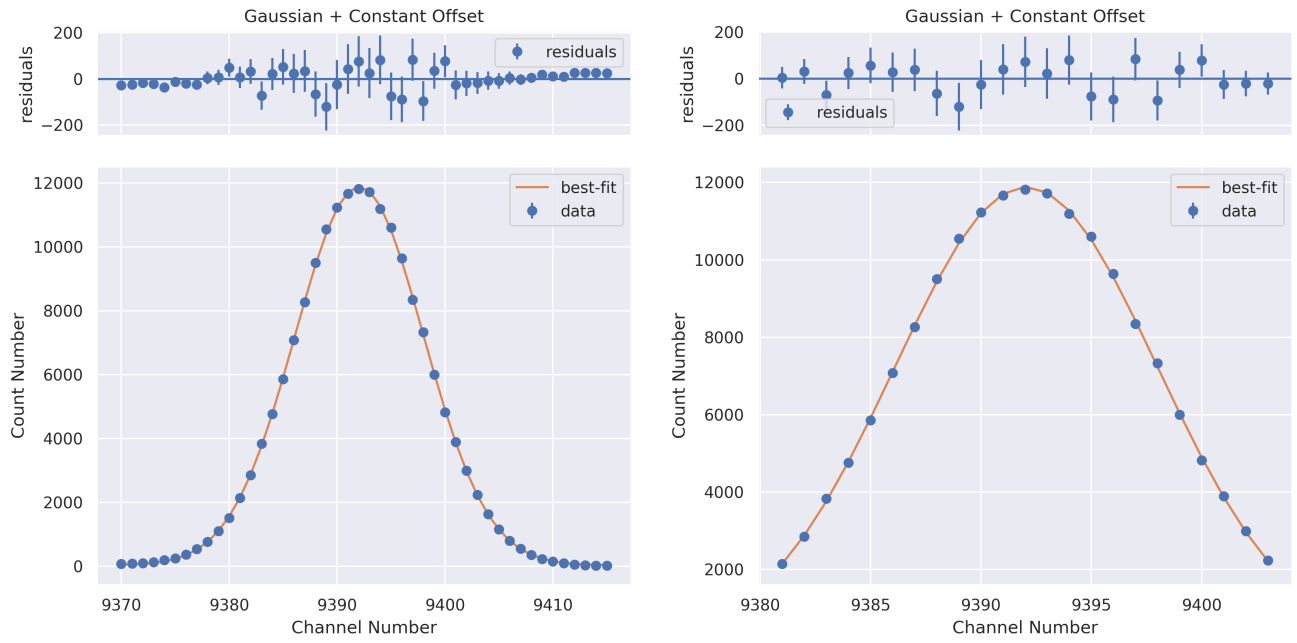


(a) Full peak with fit. $\chi^2 = 54.04$, $\chi^2_\nu = 1.39$, Prob = 5.52%, $\mu = 8269.53$, $\sigma = 5.73$



(b) Zoomed in peak with fit. $\chi^2 = 18.66$, $\chi^2_\nu = 0.93$, Prob = 54.42%, $\mu = 8269.54$, $\sigma = 5.67$

Figure A.24: Fit of full & zoomed in peak of ^{60}Co 1173 keV peak

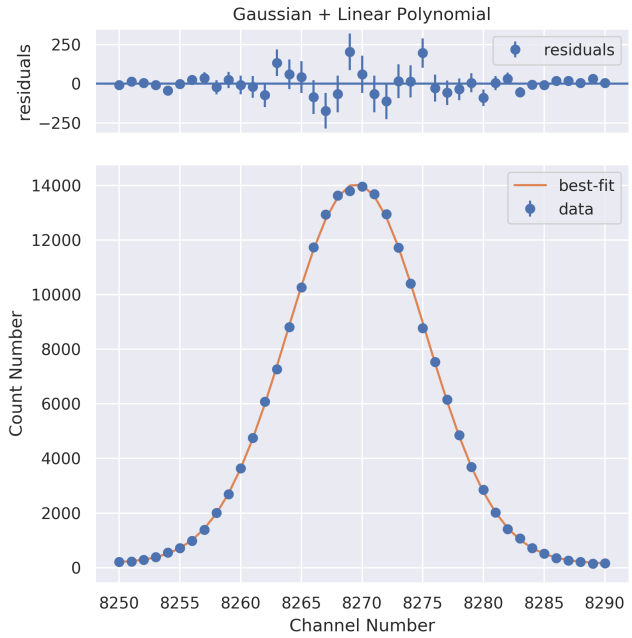


(a) Full peak with fit. $\chi^2 = 173.52$, $\chi^2_\nu = 3.94$, Prob = 0.00%, $\mu = 9392.05$, $\sigma = 5.95$

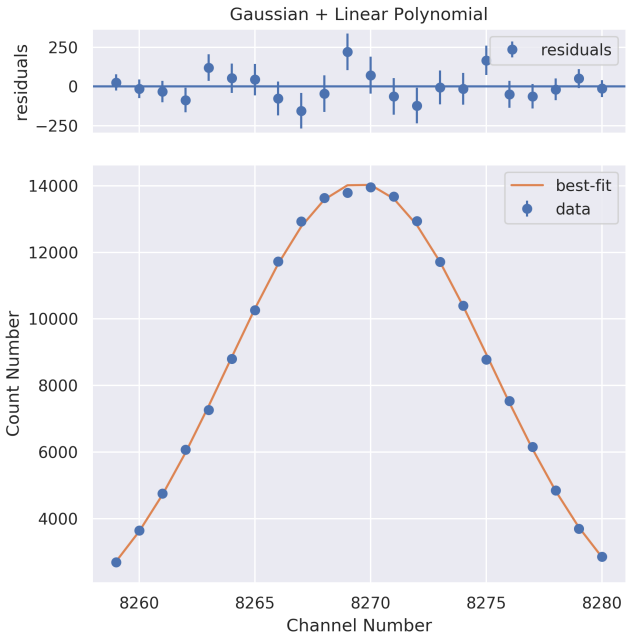
(b) Zoomed in peak with fit. $\chi^2 = 11.12$, $\chi^2_\nu = 0.53$, Prob = 96.04%, $\mu = 9392.05$, $\sigma = 5.96$

Figure A.25: Fit of full & zoomed in peak of ^{60}Co 1332 keV peak

A1.2.2 Gaussian + Linear Fit

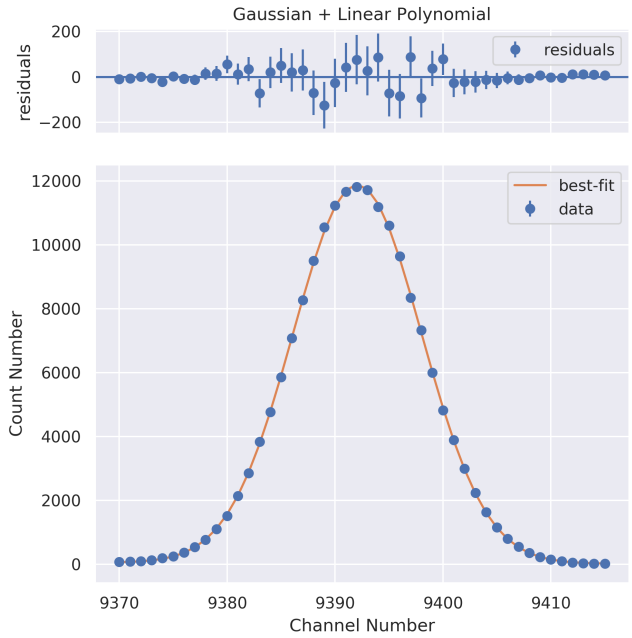


(a) Full peak with fit. $\chi^2 = 39.06$, $\chi^2_\nu = 1.00$, Prob = 46.72%, $\mu = 8269.54$, $\sigma = 5.73$

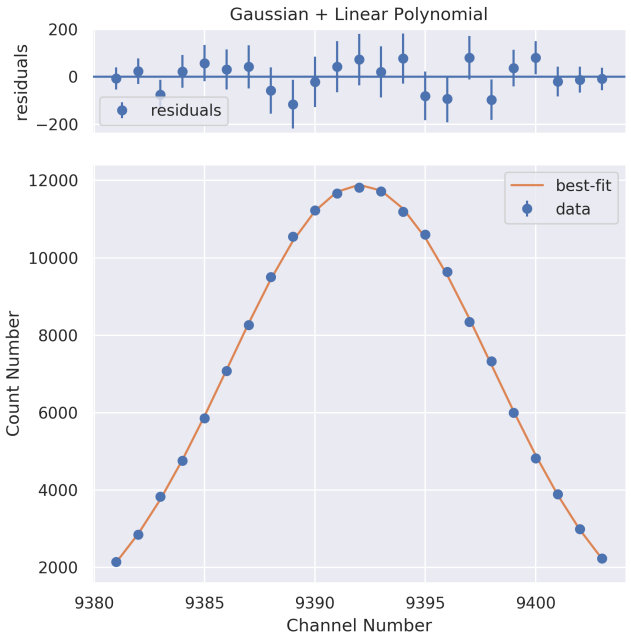


(b) Zoomed in peak with fit. $\chi^2 = 17.24$, $\chi^2_\nu = 0.86$, Prob = 63.72%, $\mu = 8269.51$, $\sigma = 5.66$

Figure A.26: Fit of full & zoomed in peak of ^{60}Co 1173 keV peak



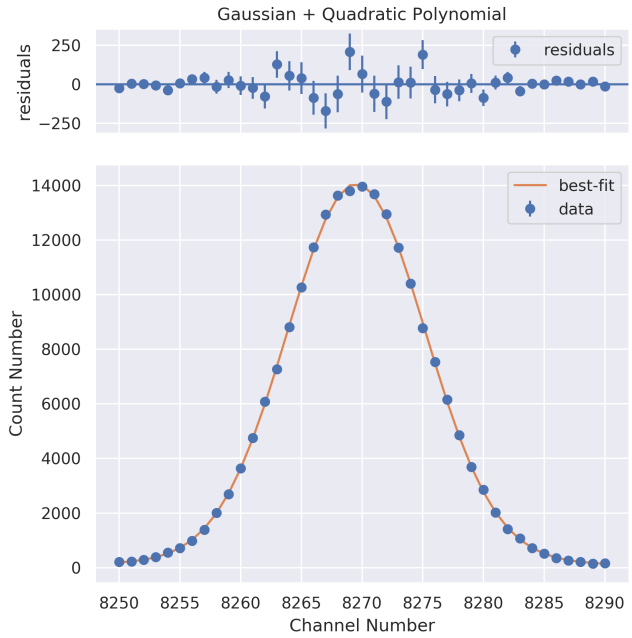
(a) Full peak with fit. $\chi^2 = 36.11$, $\chi^2_\nu = 0.82$, Prob = 79.53%, $\mu = 9392.06$, $\sigma = 5.95$



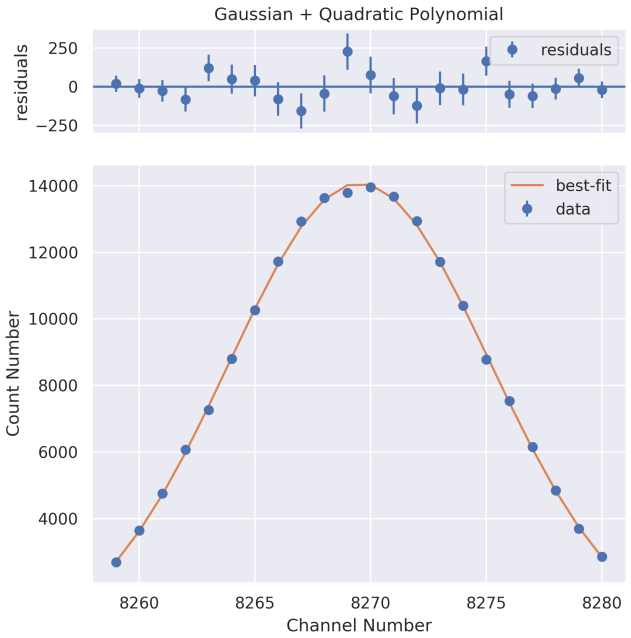
(b) Zoomed in peak with fit. $\chi^2 = 10.89$, $\chi^2_\nu = 0.52$, Prob = 96.49%, $\mu = 9392.04$, $\sigma = 5.96$

Figure A.27: Fit of full & zoomed in peak of ^{60}Co 1332 keV peak

A1.2.3 Gaussian + Quadratic Fit

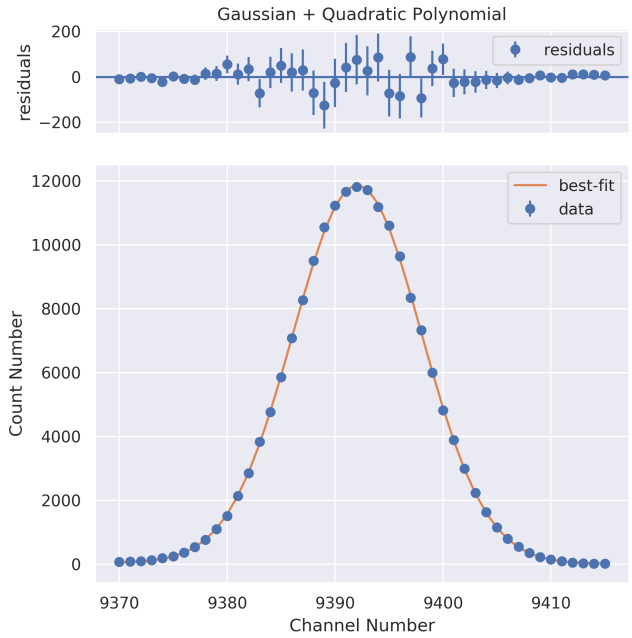


(a) Full peak with fit. $\chi^2 = 37.65$, $\chi^2_\nu = 0.97$, Prob = 53.14%, $\mu = 8269.54$, $\sigma = 5.71$

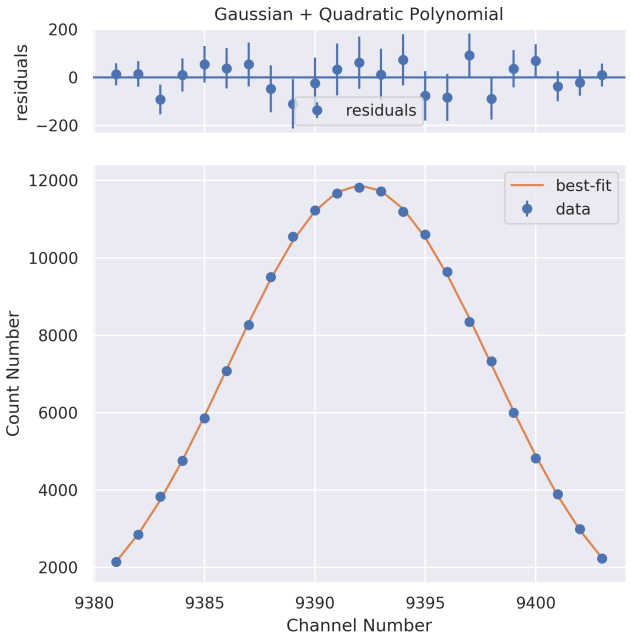


(b) Zoomed in peak with fit. $\chi^2 = 17.02$, $\chi^2_\nu = 0.85$, Prob = 65.16%, $\mu = 8269.51$, $\sigma = 5.57$

Figure A.28: Fit of full & zoomed in peak of ^{60}Co 1173 keV peak



(a) Full peak with fit. $\chi^2 = 36.03$, $\chi^2_\nu = 0.82$, Prob = 79.80%, $\mu = 9392.06$, $\sigma = 5.95$



(b) Zoomed in peak with fit. $\chi^2 = 10.95$, $\chi^2_\nu = 0.52$, Prob = 96.37%, $\mu = 9392.04$, $\sigma = 6.32$

Figure A.29: Fit of full & zoomed in peak of ^{60}Co 1332 keV peak

A1.3 Sodium-22

The ^{22}Na source had one medium energy peak, and one high-energy peak. Both of which were very distinctive, and so quite easy to fit for all functions.

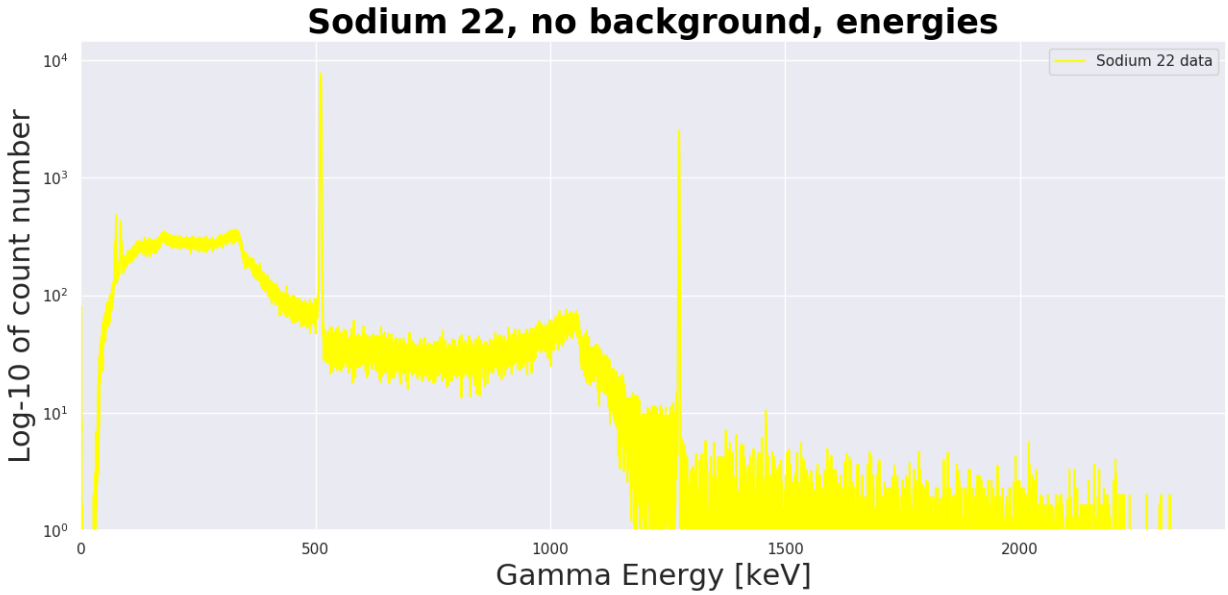
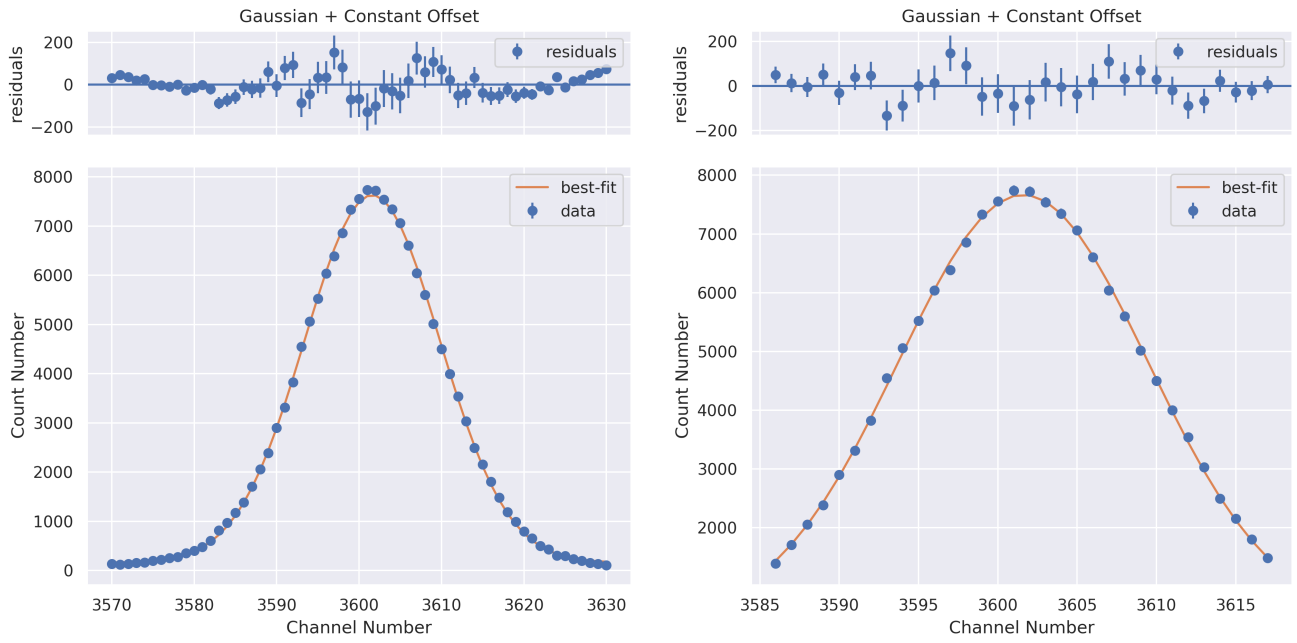


Figure A.30: The final spectrum obtained for the ^{22}Na source run for 24 hours. This has been converted to energies by using the quadratic energy calibration.

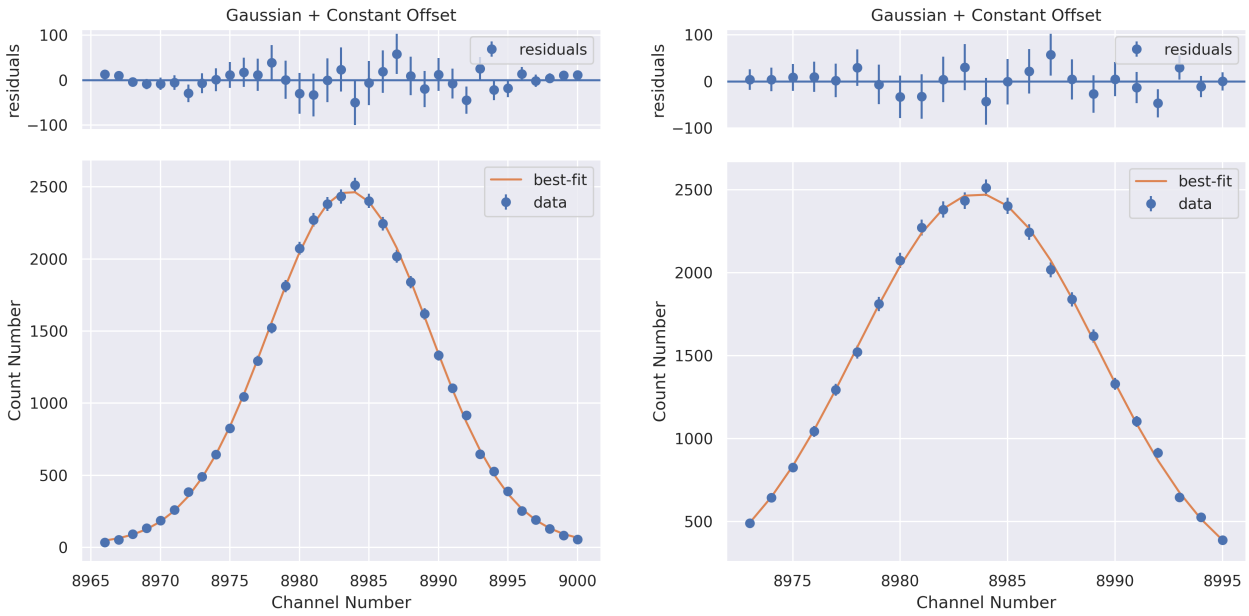
A1.3.1 Gaussian + Offset Fit



(a) Full peak with fit. $\chi^2 = 202.42$, $\chi^2_\nu = 3.43$, Prob = 0.00%, $\mu = 3601.60$, $\sigma = 8.18$

(b) Zoomed in peak with fit. $\chi^2 = 24.72$, $\chi^2_\nu = 0.82$, Prob = 73.83%, $\mu = 3601.60$, $\sigma = 7.88$

Figure A.31: Fit of full & zoomed in peak of ^{22}Na 511 keV peak

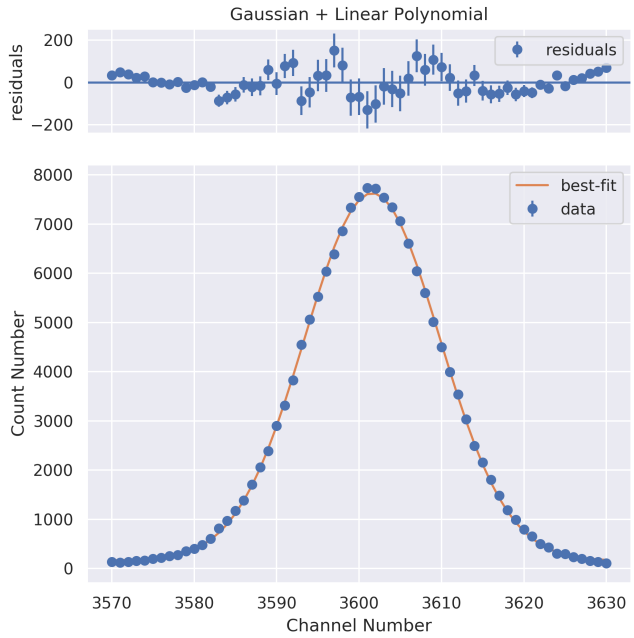


(a) Full peak with fit. $\chi^2 = 26.26$, $\chi^2_\nu = 0.80$, Prob = 79.09%, $\mu = 8983.57$, $\sigma = 5.79$

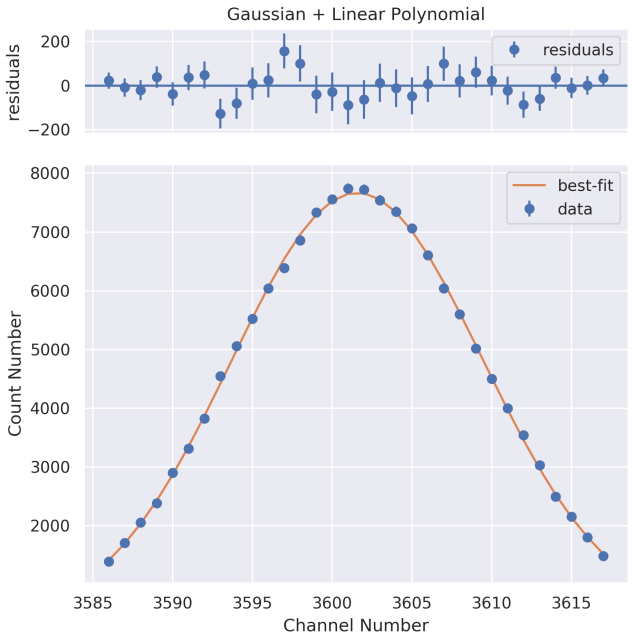
(b) Zoomed in peak with fit. $\chi^2 = 9.38$, $\chi^2_\nu = 0.45$, Prob = 98.59%, $\mu = 8983.58$, $\sigma = 5.65$

Figure A.32: Fit of full & zoomed in peak of ^{22}Na 1274 keV peak

A1.3.2 Gaussian + Linear Fit

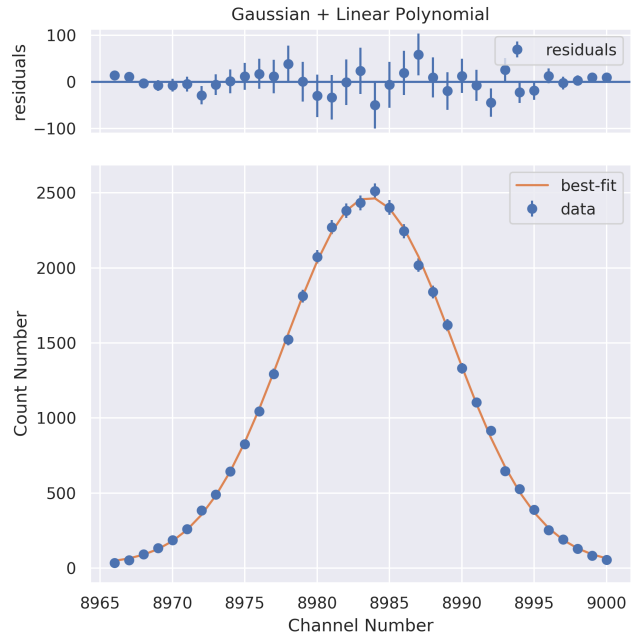


(a) Full peak with fit. $\chi^2 = 197.63$, $\chi^2_\nu = 3.35$, Prob = 0.00%, $\mu = 3601.60$, $\sigma = 8.18$

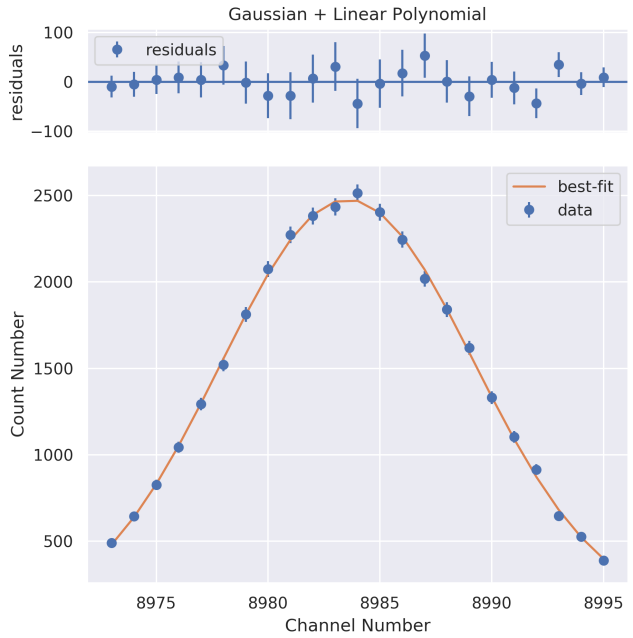


(b) Zoomed in peak with fit. $\chi^2 = 22.76$, $\chi^2_\nu = 0.76$, Prob = 82.51%, $\mu = 3601.55$, $\sigma = 7.88$

Figure A.33: Fit of full & zoomed in peak of ^{60}Na 511 keV peak



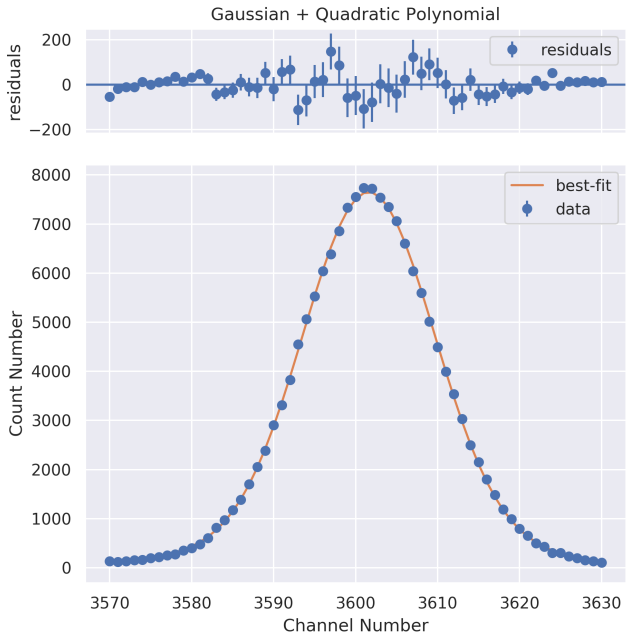
(a) Full peak with fit. $\chi^2 = 26.34$, $\chi^2_\nu = 0.80$, Prob = 78.77%, $\mu = 8983.58$, $\sigma = 5.79$



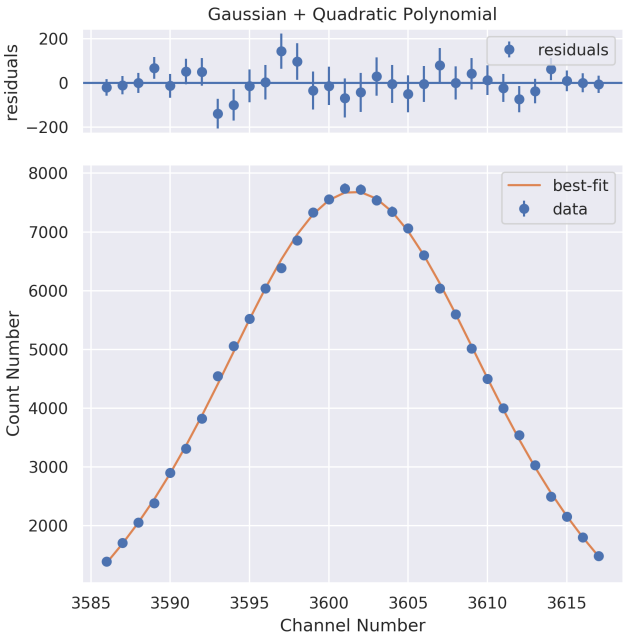
(b) Zoomed in peak with fit. $\chi^2 = 9.41$, $\chi^2_\nu = 0.45$, Prob = 98.56%, $\mu = 8983.53$, $\sigma = 5.67$

Figure A.34: Fit of full & zoomed in peak of ^{22}Na 1274 keV peak

A1.3.3 Gaussian + Quadratic Fit

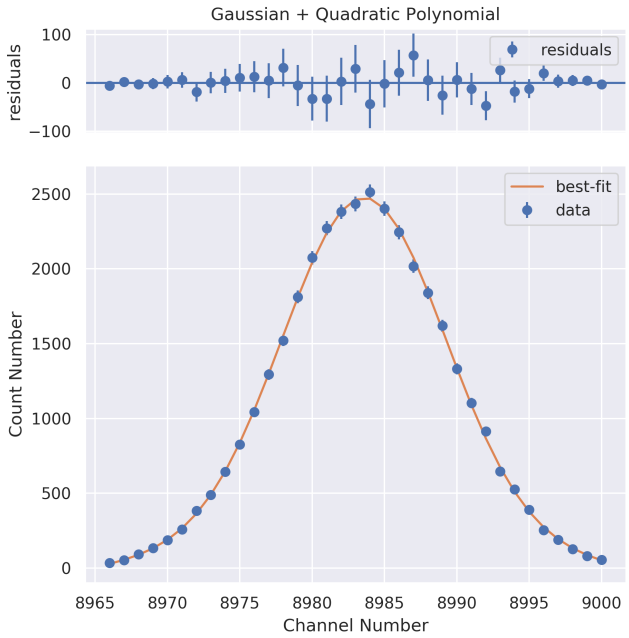


(a) Full peak with fit.$\chi^2 = 90.05$, $\chi^2_\nu = 1.53$, Prob = 0.57%, $\mu = 3601.61$, $\sigma = 7.99$

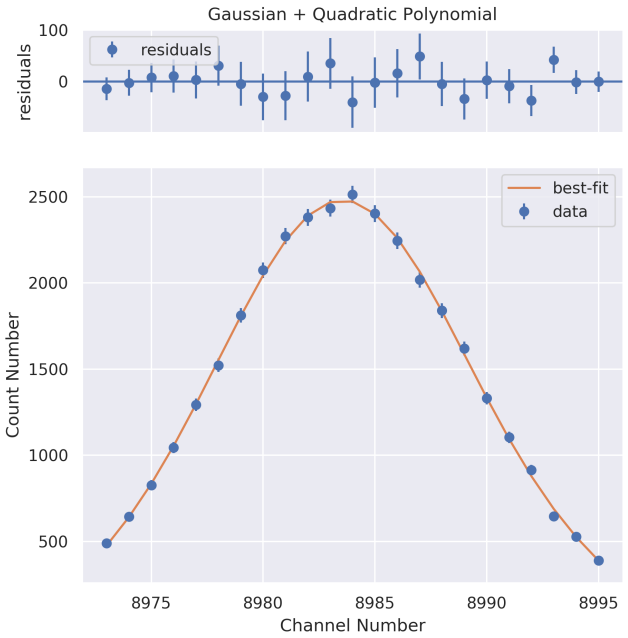


(b) Zoomed in peak with fit. $\chi^2 = 21.24$, $\chi^2_\nu = 0.71$, Prob = 88.01%, $\mu = 3601.55$, $\sigma = 7.04$

Figure A.35: Fit of full & zoomed in peak of ^{60}Na 511 keV peak



(a) Full peak with fit. $\chi^2 = 14.77$, $\chi^2_\nu = 0.45$, Prob = 99.74%, $\mu = 8983.58$, $\sigma = 5.62$



(b) Zoomed in peak with fit. $\chi^2 = 9.55$, $\chi^2_\nu = 0.45$, Prob = 98.41%, $\mu = 8983.52$, $\sigma = 5.28$

Figure A.36: Fit of full & zoomed in peak of ^{22}Na 1274 keV peak

A2 Fit Comparisons

Since there were three types of function fitted to the data, it is important that we identify which function both matched the data the best, and that is the simplest function. From functional analysis, we know that any function can be represented as an arbitrary sum of polynomials, and so by increasing the order of the fitted function will naturally cause better fit statistics, even though there is less physical reason to do so. Therefore, to find the ideal fitting function, we wanted to identify the lowest-order polynomial that correctly fitted the data, by showing that when we increase the order the fit remains the same, within errors.

This can be represented graphically by plotting the means produced for the same peaks of the different fits, with the error bars representing the standard deviation on the mean from the fit. If two error bars from different fits are overlapping with the means, then this shows that these fits are compatible with each other, and so the lowest order should be taken as the true fit. However, if they do not overlap, then this shows that they are not compatible, and so the higher order fit should be taken.

A2.1 Barium-133

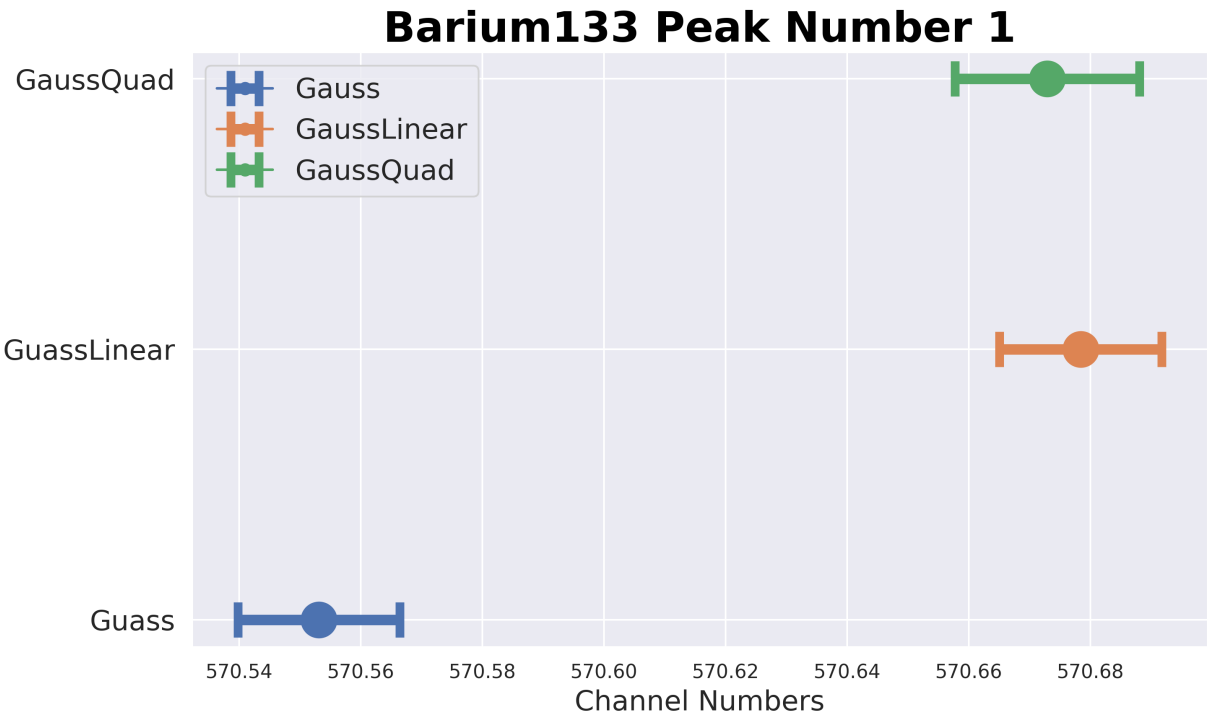


Figure A.37: Fit comparisons for ^{133}Ba 81 keV peak. Here is clear evidence that a simple Gaussian + offset fit is not sufficient, as the mean from the fit is significantly outside the acceptable area of the other two fits. This also shows that the linear polynomial provides no additional gain over the quadratic polynomial as both of their means coincide with each others error-bars, and so are statistically insignificant from each other.

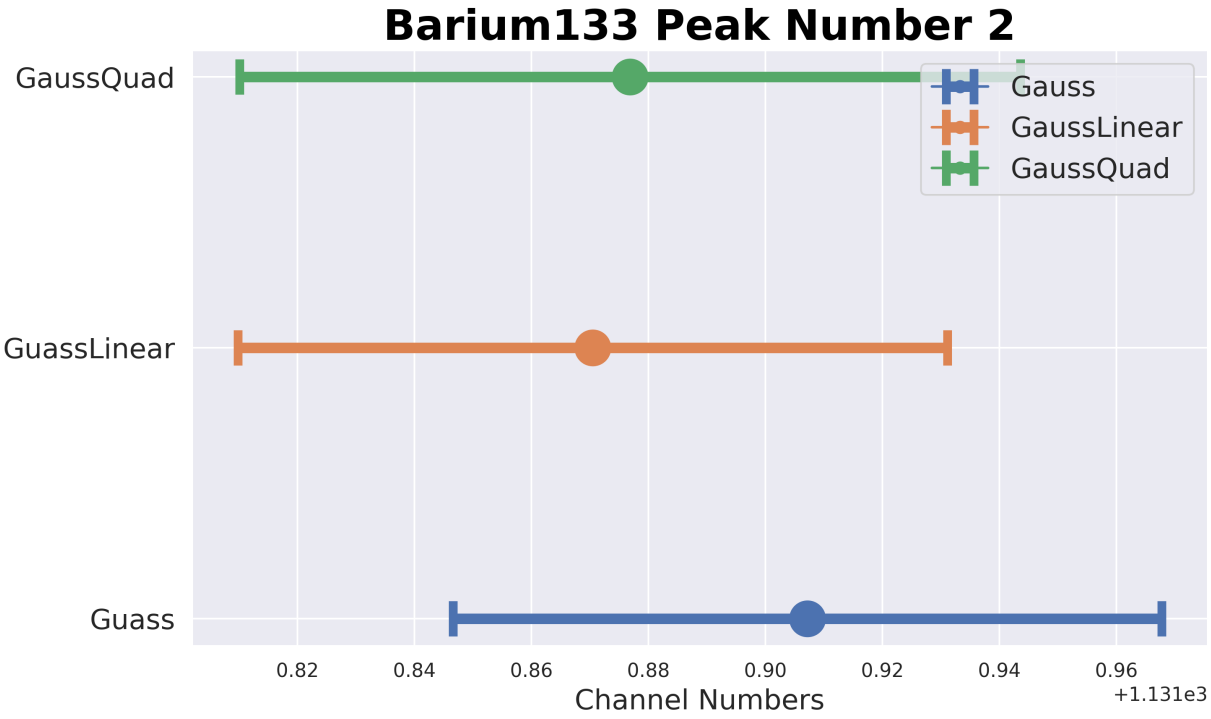


Figure A.38: Fit comparisons for ^{133}Ba 161 keV peak. Here, this shows all three means coincide with each other, and so for this particular peak going to anything above an offset has no advantage, however as other plots show that Gaussian + linear polynomial is best, that is what was used throughout.

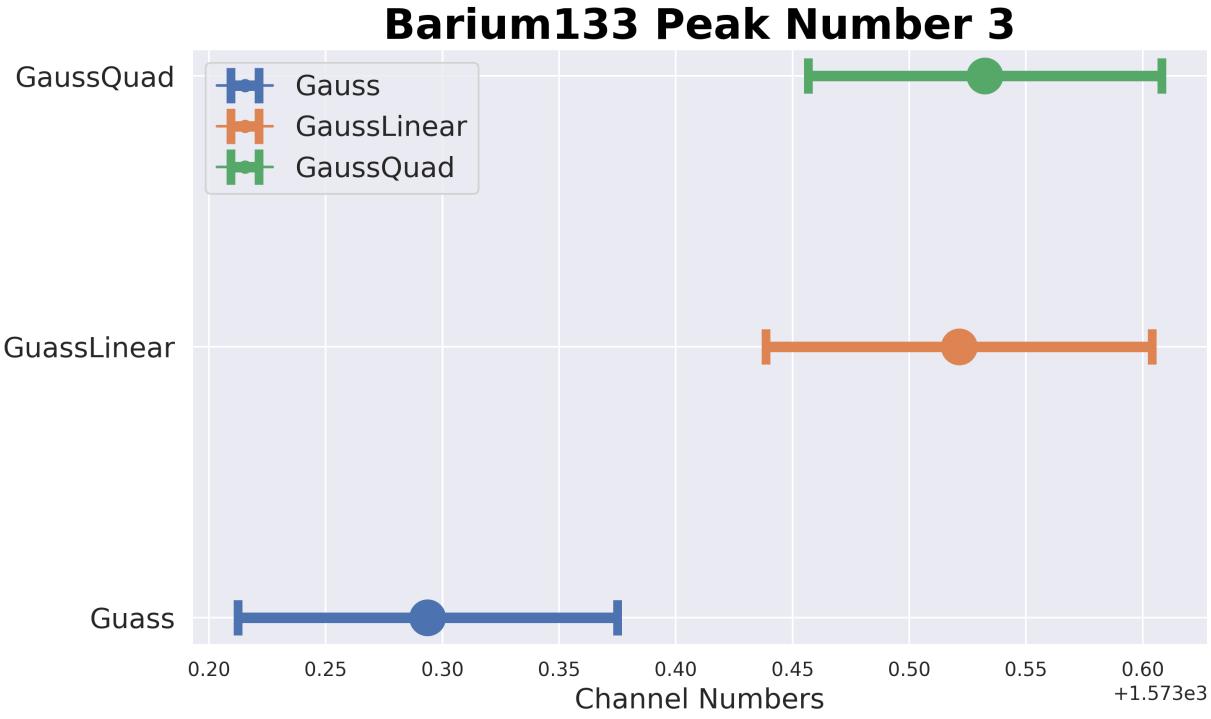


Figure A.39: Fit comparisons for ^{133}Ba 223 keV peak

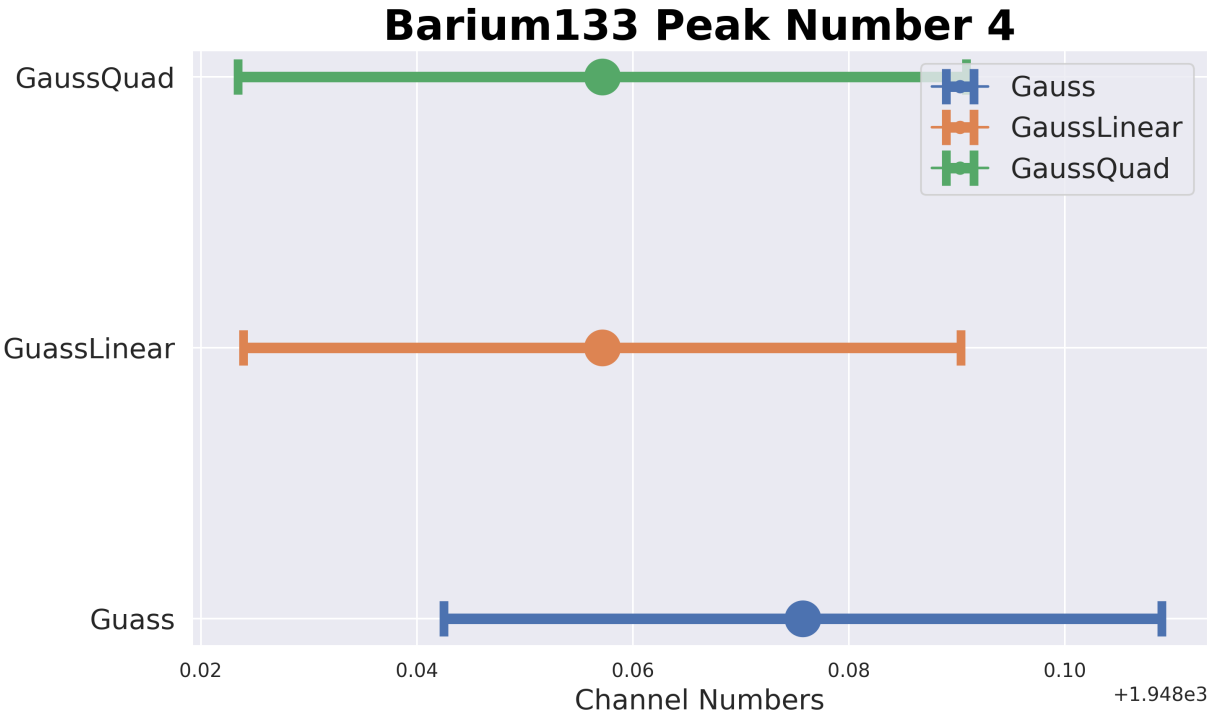


Figure A.40: Fit comparisons for ^{133}Ba 276 keV peak

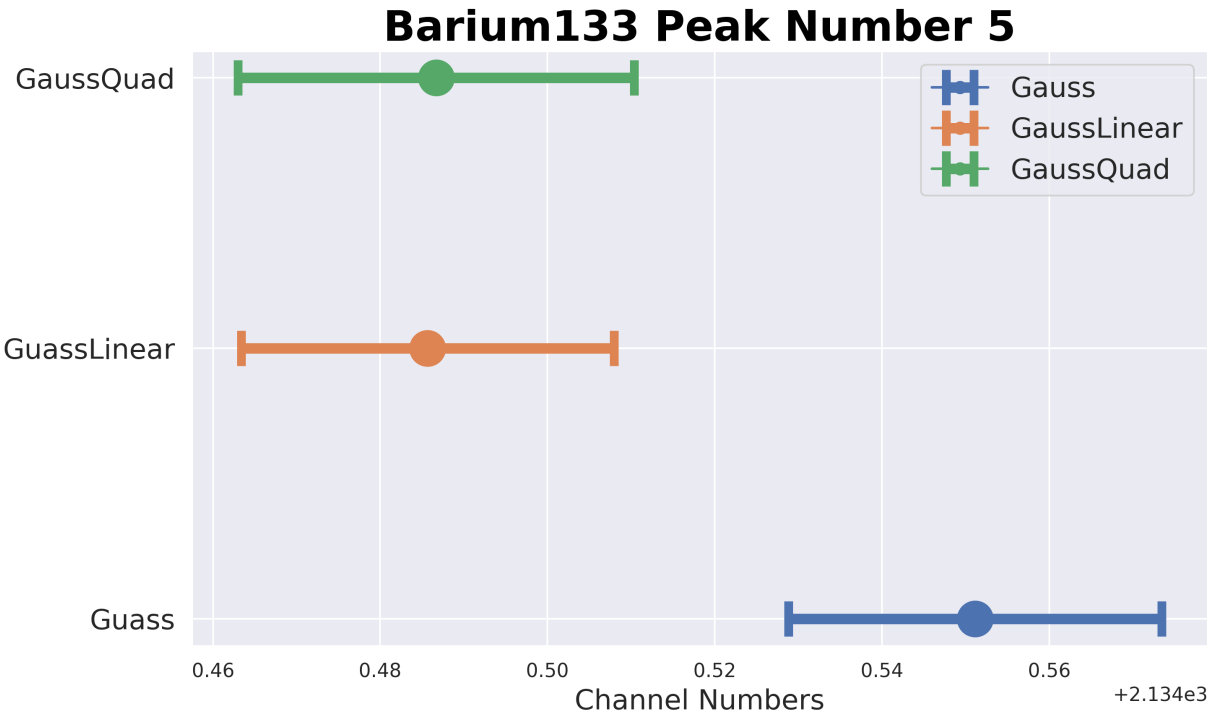


Figure A.41: Fit comparisons for ^{133}Ba 303 keV peak

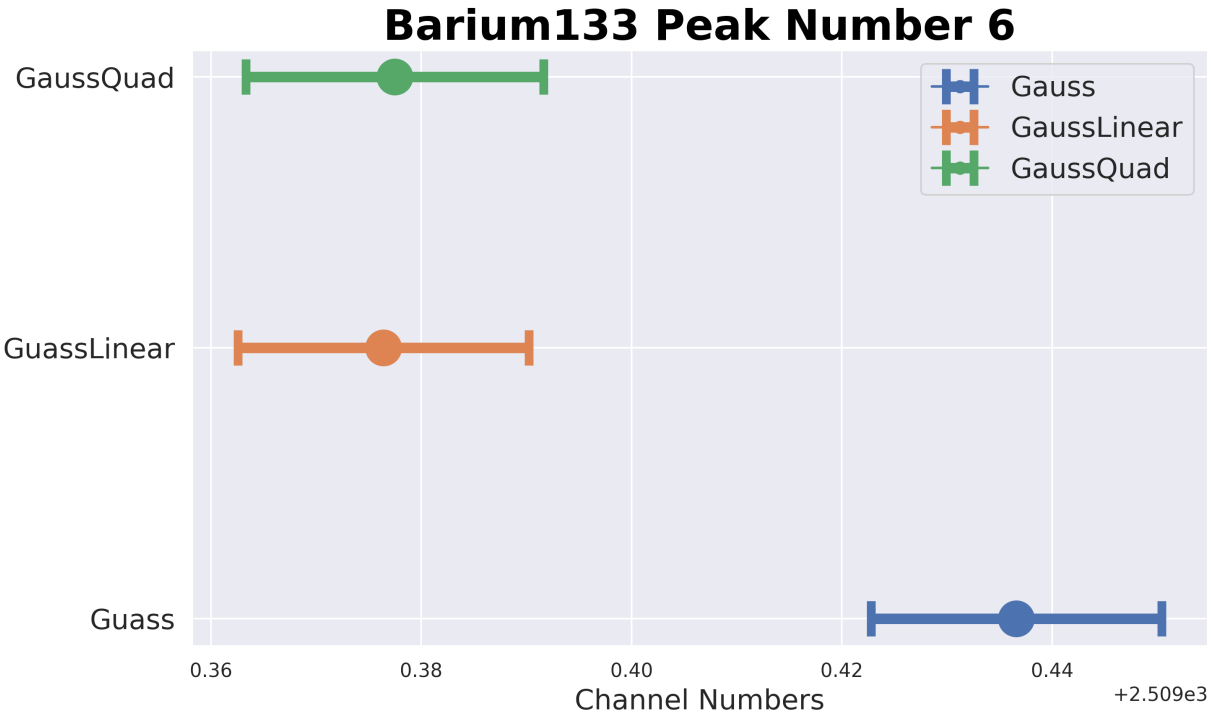


Figure A.42: Fit comparisons for ^{133}Ba 356 keV peak

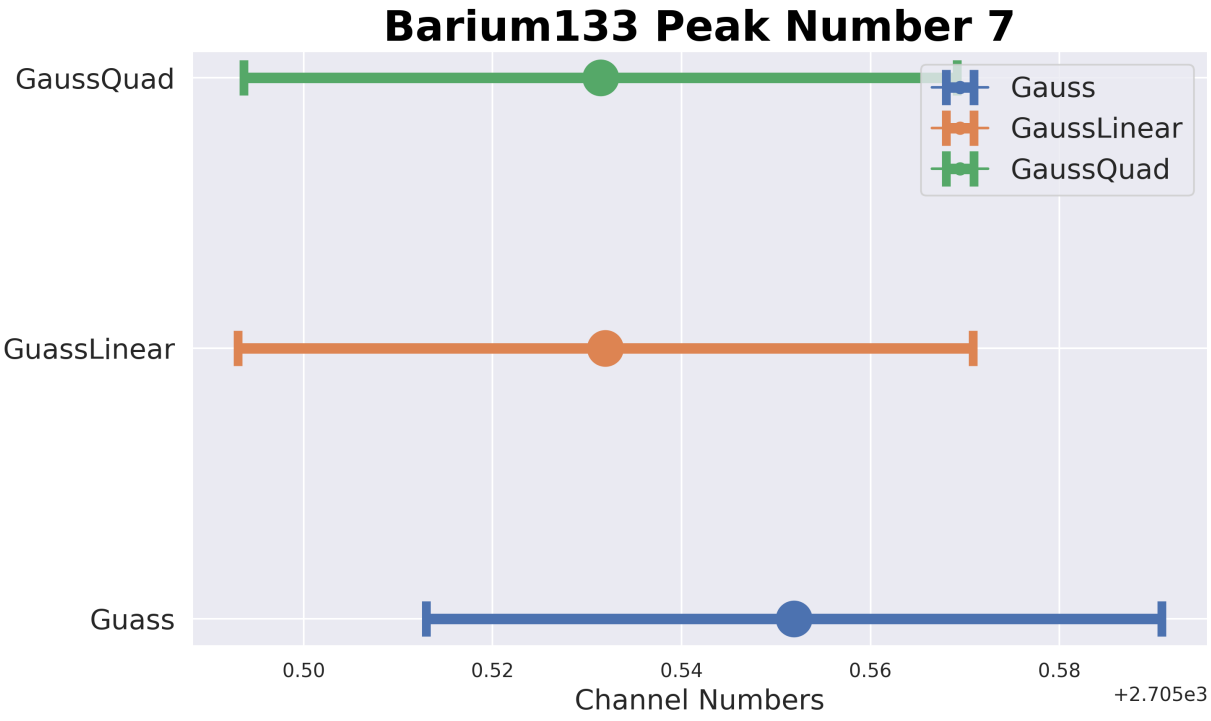


Figure A.43: Fit comparisons for ^{133}Ba 384 keV peak

A2.2 Cobalt-60

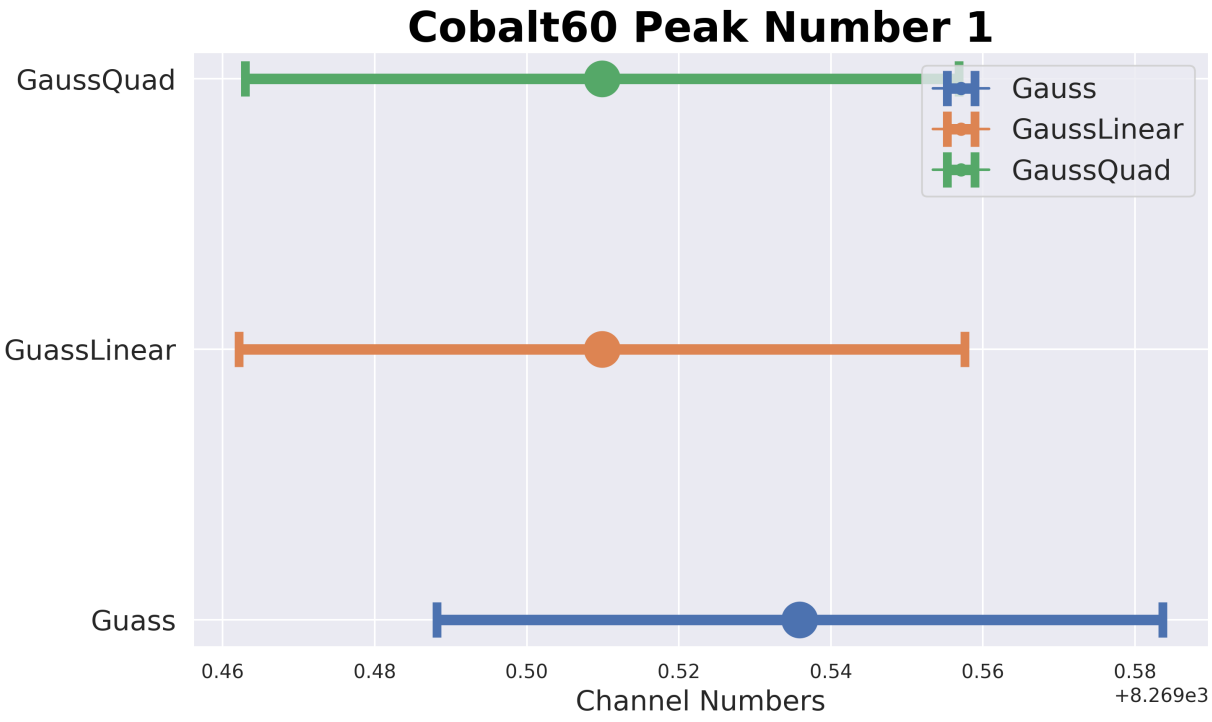


Figure A.44: Fit comparisons for ^{60}Co 1173 keV peak

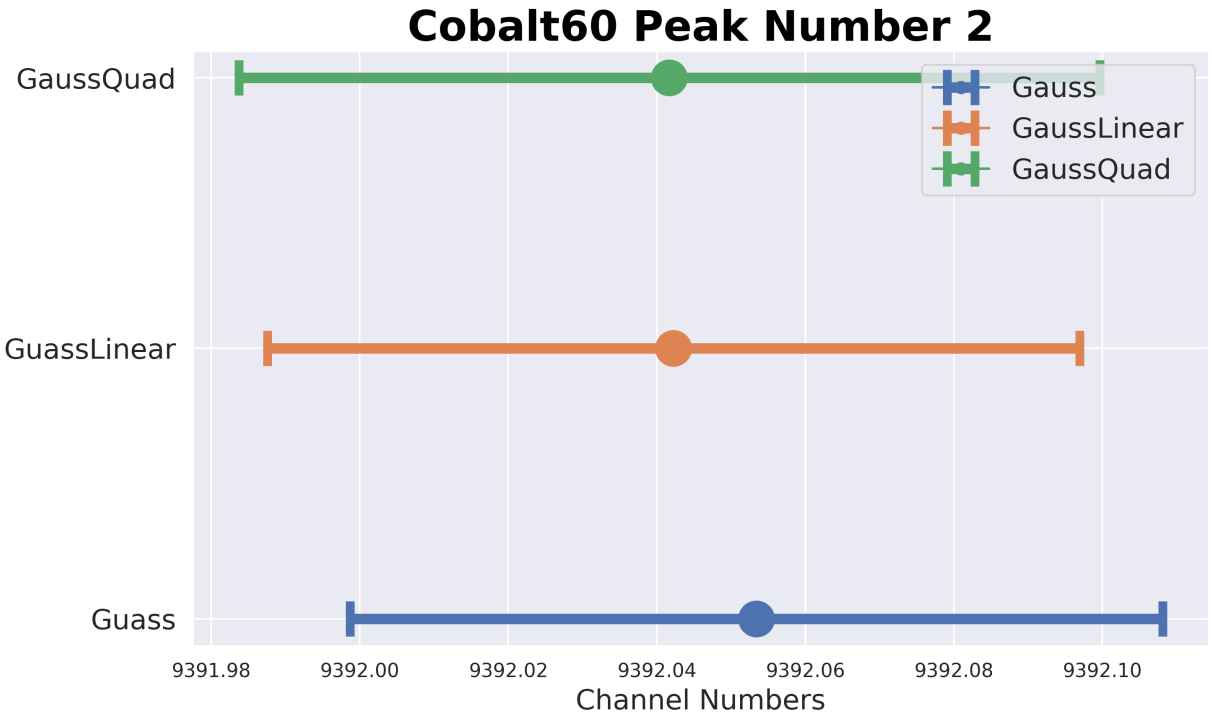


Figure A.45: Fit comparisons for ^{60}Co 1332 keV peak

A2.3 Sodium-22

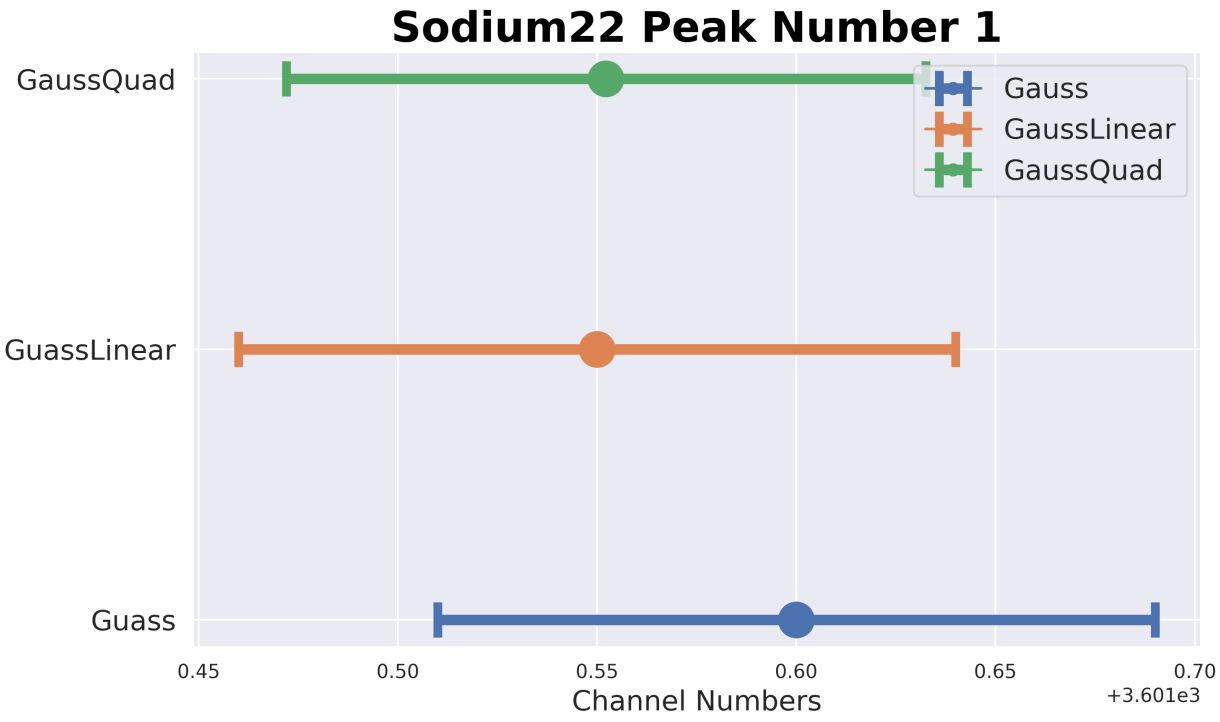


Figure A.46: Fit comparisons for ^{22}Na 511 keV peak

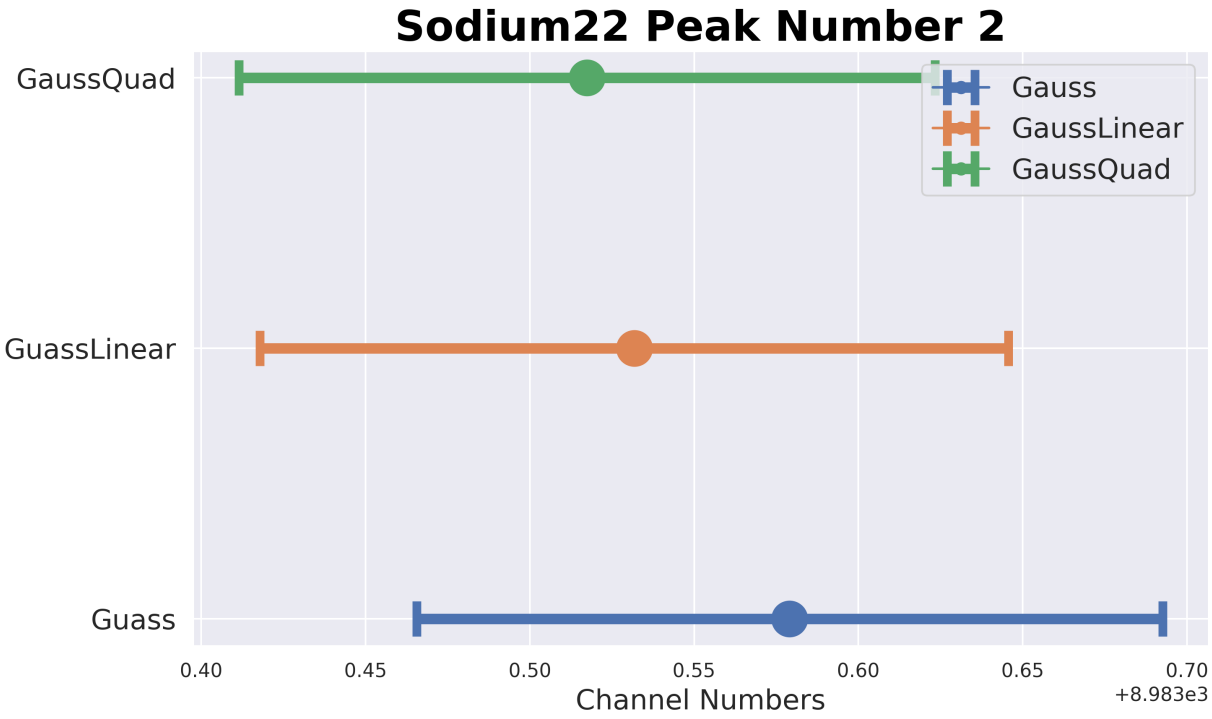


Figure A.47: Fit comparisons for ^{22}Na 1274 keV peak

A3 Python Code

Below is the complete collection of our Python codes that we wrote during the four weeks of our experiment to run various data analysis tools on the raw data, including curve fitting, energy calibration and detector resolution finding. They are included in this appendix so that way we have them on records, in case something tragic happens to the digital copies stored on the central university file system.

However, we have also mitigated against file loss by uploading our Python files, raw data, and this appendix to a GitHub repository that is hosted externally. In addition to being used as a version tracker tool helping us identify changes made to the codes over time (as we add, change & remove features), it also provides mitigation against file loss as the current builds can simply be downloaded from the repo again. This also provides an easy way for others to double check figures or results, if they so desire – improving transparency for our experiment, which is something that all scientists should be striving to do.

GitHub: `https://github.com/AlexMaraio/HPGeDetector`

A3.1 GaussianFit.py

This file is the file responsible for the fitting of the individual Gaussians to the peaks in the data. Here, we also add various order polynomials in addition to the Gaussian, starting from a simple offset to a quadratic order polynomial. This adjusts the way the fit works in subtle ways and we needed to see how fitting the various functions worked. Of course any arbitrary function can be approximated as a linear polynomial of arbitrary degree, so naturally going to a higher order polynomial will increase the goodness of fit, however detract from what can be extracted from the fit.

By looking how the mean changes between the different fit types we can determine if we are justified in increasing the order of the polynomial, and after our analysis thanks to the following code, we can not justify going beyond a linear polynomial in addition to the Gaussian function.

```
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
sns.set()
import scipy.optimize as sciopt
import scipy.stats as scistats
from lmfit import Model, Parameters
import pandas as pd

def ChiSqFunc(Measured,Fitted,Errors):
  ChiSquared = 0
  for i in range(len(Measured)):
    ChiSquared += ((Measured[i] - Fitted[i])**2.0) / ((Errors[i])**2.0)
  ReducedChiSq = ChiSquared/(len(Measured)-2)
  ProbChiSq = (1.0 - scistats.chi2.cdf(ChiSquared,len(Measured)-2) )  * 100.0
  return ChiSquared, ReducedChiSq, ProbChiSq

def Gauss(x,A,mean,sigma,a):
```

```
19    return (A/(np.sqrt(2*np.pi) * sigma )) *np.exp(-((x-mean)**2.0)/(2.0*(sigma**2.0)))
      ↪  + a
20
21  def GaussLinear(x,A,mean,sigma,a,b):
22    return (A/(np.sqrt(2*np.pi) * sigma ))*np.exp(-((x-mean)**2.0)/(2.0*(sigma**2.0)))
      ↪  + a + b*x
23
24  def GaussQuad(x,A,mean,sigma,a,b,c):
25    return (A/(np.sqrt(2*np.pi) * sigma ))*np.exp(-((x-mean)**2.0)/(2.0*(sigma**2.0)))
      ↪  + a + b*x + c*x**2.0
26
27  def ChToEnergy(Ch,a,b,c):
28    Energy = np.sqrt( ( (Ch - c + (b**2.0 / (4*a))) )/a ) - (b / (2*a))
29    return Energy
30
31  SetSigma = 2
32
33  BariumList = [ [[550,585,570,81],[1125,1140,1132,161],[1565,1582,1574,223],[1935,1960
      ↪  ,1948,276],[2118,2150,2135,303],[2493,2526,2509,356],[2688,2723,2705,384]] , []
      ↪  ]
34  SodiumList = [ [ [3570,3630,3600,511] , [8966,9000,8984,1274]] , [] ]
35  CobaltList = [ [ [8250,8290,8270,1173] , [9370,9415,9392,1332]] , [] ]
36
37  SourceList = ["Barium133-24HrRun_006_eh_1","Sodium22-24HrData_002_eh_1","Cobalt60-24H
      ↪  rData_004_eh_1"]
38
39  PlotResolution = 300
40
41  PeakNo = int(1)
42
43  datadict = {'Element':[],'Fit type':[], 'Peak number':[], 'Peak type':[], 'Energy
      ↪  (keV)': [], 'Resolution':[], 'Min of range':[], 'Max of range':[], 'Mean':[],
      ↪  'A':[], 'Sigma':[], 'Error on mean':[], 'a':[], 'b':[], 'c':[], 'chisq':[],
      ↪  'Reduced chisq':[], 'Probchisq':[]}
44
45  # Fit parameters
46  a = 2.63977565e-06
47  b = 7.04767648
48  c = -1.91414134e-01
49
50  for source in SourceList:
51    if source == "Barium133-24HrRun_006_eh_1":
52      ElementList = BariumList
53      Element = "Barium133"
54    elif source == "Sodium22-24HrData_002_eh_1":
55      ElementList = SodiumList
56      Element = "Sodium22"
57    else:
58      ElementList = CobaltList
59      Element = "Cobalt60"
60
61    for item in ElementList:
62      for peak in item:
```

```
63        #! Initialises the source and ranges for run
64
65        if item == ElementList[0]:
66          PeakType = "Full"
67        else:
68          PeakType = "Zoom"
69
70        df = pd.read_csv(source + ".dat", sep = r"\s+", names = ['channel
          ↪  number','count number'])
71
72        df['count errors'] = np.sqrt(df['count number'])
73        df['count errors'] = df['count errors'].replace(0,1)
74
75        MinValue = peak[0]
76        MaxValue = peak[1]
77        MeanValue = peak[2]
78        PeakEnergy = peak[3]
79
80        data = df[(MinValue<=df['channel number']) & (df['channel number']<=MaxValue)]
81        #------------------------------------------------------------------------
82        #! Gaussian + Offset model only
83
84        GaussModel = Model(Gauss)
85        Params = Parameters()
86
87        Params.add('A',value=max(data['count number']),vary=True,min=0)
88        Params.add('mean',value=MeanValue,vary=True)
89        Params.add('sigma',value=1,vary=True)
90        Params.add('a',value=1,vary=True)
91
92        FitResult = GaussModel.fit(data['count number'],params=Params,x=data['channel
          ↪  number'])
93        TempList = [ FitResult.best_values['mean'] - SetSigma
          ↪  *FitResult.best_values['sigma'] , FitResult.best_values['mean'] + SetSigma
          ↪  *FitResult.best_values['sigma'] , FitResult.best_values['mean'] ,
          ↪  PeakEnergy ]
94        if item == ElementList[0]:
95          ElementList[1].append(TempList)
96        #print(FitResult.best_values)
97        FitResult.plot(yerr=data['count errors'],xlabel='Channel Number',ylabel='Count
          ↪  Number',title='Gaussian + Constant Offset')
98
99        ChiStats = ChiSqFunc(list(data['count
          ↪  number']),list(FitResult.best_fit),list(data['count errors']))
100
101       plt.tight_layout()
102       plt.savefig(f'Plots/{Element}/Gauss/Gauss_{PeakNo}_{PeakType}.png',
          ↪  format='png', dpi=PlotResolution)
103       plt.close('all')
104
105       CountMax1 = FitResult.best_values['A'] / ( FitResult.best_values['sigma'] *
          ↪  np.sqrt(2 * np.pi ) ) + FitResult.best_values['a']
106       ErrorOnMean1 = FitResult.best_values['sigma'] / np.sqrt(CountMax1)
```

```
107
108         datadict['Element'].append(Element)
109         datadict['Fit type'].append('Gauss')
110         datadict['Energy (keV)'].append(PeakEnergy)
111         FWHM_Energy = 2 * np.sqrt(2*np.log(2)) *
            ↪ ChToEnergy(FitResult.best_values['sigma'],a,b,c)
112         datadict['Resolution'].append( FWHM_Energy / PeakEnergy)
113         #datadict['Resolution'].append( ( (2 *
            ↪ np.sqrt(2*np.log(2))*FitResult.best_values['sigma'] - b ) / a ) /
            ↪ PeakEnergy)
114         datadict['Peak number'].append(PeakNo)
115         datadict['Peak type'].append(PeakType)
116         datadict['Min of range'].append(MinValue)
117         datadict['Max of range'].append(MaxValue)
118         datadict['Mean'].append(FitResult.best_values['mean'])
119         datadict['A'].append(FitResult.best_values['A'])
120         datadict['Sigma'].append(FitResult.best_values['sigma'])
121         datadict['Error on mean'].append(1)
122         datadict['a'].append(FitResult.best_values['a'])
123         datadict['b'].append(0)
124         datadict['c'].append(0)
125         datadict['chisq'].append(ChiStats[0])
126         datadict['Reduced chisq'].append(ChiStats[1])
127         datadict['Probchisq'].append(ChiStats[2])
128
129
130         #-------------------------------------------------------------------------
131         #! Gaussain + Linear model
132
133         GaussModel2 = Model(GaussLinear)
134         Params2 = Parameters()
135
136         Params2.add('A',value=max(data['count number']),vary=True,min=0)
137         Params2.add('mean',value=MeanValue,vary=True)
138         Params2.add('sigma',value=1,vary=True)
139         Params2.add('a',value=1,vary=True)
140         Params2.add('b',value=1,vary=True)
141
142         FitResult2 = GaussModel2.fit(data['count
            ↪ number'],params=Params2,x=data['channel number'])
143         FitResult2.plot(yerr=data['count errors'],xlabel='Channel Number',ylabel='Count
            ↪ Number',title='Gaussian + Linear Polynomial')
144
145         ChiStats2 = ChiSqFunc(list(data['count
            ↪ number']),list(FitResult2.best_fit),list(data['count errors']))
146
147         plt.tight_layout()
148         plt.savefig(f'Plots/{Element}/Linear/Linear_{PeakNo}_{PeakType}.png',
            ↪ format='png', dpi=PlotResolution)
149         plt.close('all')
150
151         CountMax2 = FitResult2.best_values['A'] / ( FitResult2.best_values['sigma'] *
            ↪ np.sqrt(2 * np.pi ) ) + FitResult2.best_values['a'] +
            ↪ FitResult2.best_values['b'] * FitResult2.best_values['mean']
```

```
152         ErrorOnMean2 = FitResult2.best_values['sigma'] / np.sqrt(CountMax2)
153 
154         datadict['Element'].append(Element)
155         datadict['Fit type'].append('Linear')
156         datadict['Energy (keV)'].append(PeakEnergy)
157         FWHM_Energy = 2 * np.sqrt(2*np.log(2)) *
            ↪  ChToEnergy(FitResult2.best_values['sigma'],a,b,c)
158         datadict['Resolution'].append( FWHM_Energy / PeakEnergy)
159         datadict['Peak number'].append(PeakNo)
160         datadict['Peak type'].append(PeakType)
161         datadict['Min of range'].append(MinValue)
162         datadict['Max of range'].append(MaxValue)
163         datadict['Mean'].append(FitResult2.best_values['mean'])
164         datadict['A'].append(FitResult2.best_values['A'])
165         datadict['Sigma'].append(FitResult2.best_values['sigma'])
166         datadict['Error on mean'].append(1)
167         datadict['a'].append(FitResult2.best_values['a'])
168         datadict['b'].append(FitResult2.best_values['b'])
169         datadict['c'].append(0)
170         datadict['chisq'].append(ChiStats2[0])
171         datadict['Reduced chisq'].append(ChiStats2[1])
172         datadict['Probchisq'].append(ChiStats2[2])
173 
174         #-------------------------------------------------------------------------
175         #! Gaussain + Quadratic model
176 
177         GaussModel3 = Model(GaussQuad)
178         Params3 = Parameters()
179 
180         Params3.add('A',value=max(data['count number']),vary=True,min=0)
181         Params3.add('mean',value=MeanValue,vary=True)
182         Params3.add('sigma',value=1,vary=True)
183         Params3.add('a',value=1,vary=True)
184         Params3.add('b',value=1,vary=True)
185         Params3.add('c',value=1,vary=True)
186 
187         FitResult3 = GaussModel3.fit(data['count
            ↪  number'],params=Params3,x=data['channel number'])
188         FitResult3.plot(yerr=data['count errors'],xlabel='Channel Number',ylabel='Count
            ↪  Number',title='Gaussian + Quadratic Polynomial')
189 
190         ChiStats3 = ChiSqFunc(list(data['count
            ↪  number']),list(FitResult3.best_fit),list(data['count errors']))
191 
192         plt.tight_layout()
193         plt.savefig(f'Plots/{Element}/Quad/Quad_{PeakNo}_{PeakType}.png', format='png',
            ↪  dpi=PlotResolution)
194         plt.close('all')
195 
196         CountMax3 = FitResult3.best_values['A'] / ( FitResult3.best_values['sigma'] *
            ↪  np.sqrt(2 * np.pi ) ) + FitResult3.best_values['a'] +
            ↪  FitResult3.best_values['b'] * FitResult3.best_values['mean'] +
            ↪  FitResult3.best_values['c'] * (FitResult3.best_values['mean'])** 2.0
```

```
197         ErrorOnMean3 = FitResult3.best_values['sigma'] / np.sqrt(CountMax3)
198 
199         datadict['Element'].append(Element)
200         datadict['Fit type'].append('Quad')
201         datadict['Energy (keV)'].append(PeakEnergy)
202         FWHM_Energy = 2 * np.sqrt(2*np.log(2)) *
            ↪  ChToEnergy(FitResult3.best_values['sigma'],a,b,c)
203         datadict['Resolution'].append( FWHM_Energy / PeakEnergy)
204         datadict['Peak number'].append(PeakNo)
205         datadict['Peak type'].append(PeakType)
206         datadict['Min of range'].append(MinValue)
207         datadict['Max of range'].append(MaxValue)
208         datadict['Mean'].append(FitResult3.best_values['mean'])
209         datadict['A'].append(FitResult3.best_values['A'])
210         datadict['Sigma'].append(FitResult3.best_values['sigma'])
211         datadict['Error on mean'].append(1)
212         datadict['a'].append(FitResult3.best_values['a'])
213         datadict['b'].append(FitResult3.best_values['b'])
214         datadict['c'].append(FitResult3.best_values['c'])
215         datadict['chisq'].append(ChiStats3[0])
216         datadict['Reduced chisq'].append(ChiStats3[1])
217         datadict['Probchisq'].append(ChiStats3[2])
218 
219 
220         Delta12 = np.abs(FitResult.best_values['mean'] - FitResult2.best_values['mean'])
221         ErrorDelta12 = np.sqrt( ErrorOnMean1**2.0 + ErrorOnMean2**2.0 )
222 
223         Delta23 = np.abs(FitResult2.best_values['mean'] -
            ↪  FitResult3.best_values['mean'])
224         ErrorDelta23 = np.sqrt( ErrorOnMean2**2.0 + ErrorOnMean3**2.0 )
225 
226 
227         if PeakType == "Zoom":
228           if abs(ErrorDelta23) > abs(Delta23):
229             print('best fit')
230           else:
231             print('not best fit')
232 
233           TitleFont = {'size':'24', 'color':'black', 'weight':'bold'}
234           AxTitleFont = {'size':'16'}
235           plt.figure(figsize=(10,6))
236           plt.errorbar(FitResult.best_values['mean'],1,xerr=ErrorOnMean1,elinewidth=6,c
              ↪  apsize=10,capthick=5,marker='o',markersize=20,label="Gauss")
237           plt.errorbar(FitResult2.best_values['mean'],1.25,xerr=ErrorOnMean2,elinewidth
              ↪  =6,capsize=10,capthick=5,marker='o',markersize=20,label="GaussLinear")
238           plt.errorbar(FitResult3.best_values['mean'],1.5,xerr=ErrorOnMean3,elinewidth=
              ↪  6,capsize=10,capthick=5,marker='o',markersize=20,label="GaussQuad")
239           plt.legend(fontsize=16,markerscale=0.33)
240           plt.title(str(Element) + ' Peak Number ' + str(PeakNo),**TitleFont)
241           plt.xlabel('Channel Numbers',**AxTitleFont)
242           plt.yticks([1,1.25,1.5],['Guass','GuassLinear','GaussQuad'],**AxTitleFont)
243           plt.tight_layout()
244           plt.savefig(f'Plots/{Element}/FitComparison_Peak{PeakNo}.png', format='png',
              ↪  dpi=PlotResolution)
```

```
245             plt.close('all')
246 
247         PeakNo +=1
248 
249     PeakNo = 1
250 
251 Fitdf = pd.DataFrame(datadict)
252 Fitdf.to_csv('Gamma_Peak_Stats_and_Params.csv')
```

A3.2 Calibration.py

This Python file was responsible for reading in the fitted Gaussians that were generated by `GaussainFit.py` and turning them into the fit between channel number and energy. First, we used a linear fit as it was assumed that the equipment would have a linear response, however, it was observed that for large energies, there was a large discrepancy between our measured value and the predicted value for the peaks. This lead us to believe that the energy calibration might have to be done with a quadratic function, not just a simple straight line.

When we performed this new quadratic fit, we found that the energies matched much more closely to what we expected them to be, and so we can determine that the system overall has a non-linear response between energy and channel number.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as sciopt
import seaborn as sns
import pandas as pd
sns.set()
import scipy.stats as scistats

TitleFont = {'size':'25', 'color':'black', 'weight':'bold'}
AxTitleFont = {'size':'22'}

def Linear(E,a,b):
    return E*a + b

def Quadratic(E,a,b,c):
    return a*E**2.0 + E*b + c

def ChiSqFunc(Measured,Fitted,Errors,Params):
    ChiSquared = 0
    for i in range(len(Measured)):
        ChiSquared += ((Measured[i] - Fitted[i])**2.0) / ((Errors[i])**2.0)
    ReducedChiSq = ChiSquared/(len(Measured)-len(Params))
    ProbChiSq = (1.0 - scistats.chi2.cdf(ChiSquared,len(Measured)-len(Params)) )  *
     ↪  100.0
    return ChiSquared, ReducedChiSq, ProbChiSq

#? Imports the csv to a Pandas DataFrame
FitDF = pd.read_csv('Gamma_Peak_Stats_and_Params.csv', names=['Element', 'Fit type',
 ↪  'Peak number',  'Peak type', 'Energy (keV)', 'Resolution', 'Min', 'Max', 'Mean',
 ↪  'A', 'Sigma', 'Error' ,'a', 'b', 'chisq', 'Reduced chisq',
 ↪  'Probchisq'],usecols=list(range(1,18)))
#? Ensures that only the linear and zoomed peaks are fitted
FitDF = FitDF[FitDF['Fit type'] == 'Linear']
FitDF = FitDF[FitDF['Peak type'] == 'Zoom']
#? Puts the DF as floats otherwise it crashes
FitDF['Mean'] = FitDF['Mean'].astype(float)
FitDF['Error'] = FitDF['Error'].astype(float)
FitDF['Energy (keV)'] = FitDF['Energy (keV)'].astype(float)

```

```
DataBa = FitDF[FitDF['Element'] == 'Barium133']
DataCo = FitDF[FitDF['Element'] == 'Cobalt60']
DataNa = FitDF[FitDF['Element'] == 'Sodium22']

#? Fits the quadratic to the data
ParamsLinear, ErrorsLinear = sciopt.curve_fit(Quadratic,FitDF['Energy (keV)'],FitDF['Mean'])
print(ParamsLinear)

a = ParamsLinear[0]
b = ParamsLinear[1]
c = ParamsLinear[2]

Energies = np.sqrt( (FitDF['Mean'] - c + (b**2.0/ (4 * a)))/a ) - (b/(2*a))
#Energies = (FitDF['Mean'] - b)/(a) # keV

#? Calibration Plot
plt.errorbar(DataBa['Energy (keV)'],DataBa['Mean'], fmt='ro',label="Barium-133", markersize=10.5,yerr=DataBa['Error'].values)
plt.errorbar(DataCo['Energy (keV)'],DataCo['Mean'],fmt='bo',label="Cobalt-60", markersize=10.5,yerr=DataCo['Error'].values)
plt.errorbar(DataNa['Energy (keV)'],DataNa['Mean'],fmt='yo',label="Sodium-22", markersize=10.5,yerr=DataNa['Error'].values)
plt.plot(Energies,Quadratic(Energies,*ParamsLinear),label='Fit')
plt.title('Energy calibration plot with fit',**TitleFont)
plt.xlabel('Gamma Energy [keV]',**AxTitleFont)
plt.ylabel('Channel Number',**AxTitleFont)
plt.legend()
plt.show()

#? Residual plot
plt.errorbar(DataBa['Energy (keV)'],[(DataBa['Mean'].iloc[i] - Quadratic(DataBa['Energy (keV)'].iloc[i],*ParamsLinear)) for i in range(len(DataBa['Energy (keV)']))],fmt='ro',label="Barium-133", markersize=7.5,yerr=DataBa['Error'].values)
plt.errorbar(DataCo['Energy (keV)'],[(DataCo['Mean'].iloc[i] - Quadratic(DataCo['Energy (keV)'].iloc[i],*ParamsLinear)) for i in range(len(DataCo['Energy (keV)']))],fmt='bo',label="Cobalt-60", markersize=7.5,yerr=DataCo['Error'].values)
plt.errorbar(DataNa['Energy (keV)'],[(DataNa['Mean'].iloc[i] - Quadratic(DataNa['Energy (keV)'].iloc[i],*ParamsLinear)) for i in range(len(DataNa['Energy (keV)']))],fmt='yo',label="Sodium-22", markersize=7.5,yerr=DataNa['Error'].values)
plt.title('Energy calibration residual plot',**TitleFont)
plt.xlabel('Gamma Energy [keV]',**AxTitleFont)
plt.ylabel('Channel Number Residual',**AxTitleFont)
plt.legend()
plt.show()

ChiStats = ChiSqFunc(list(FitDF['Mean']),list(Quadratic(FitDF['Energy (keV)'],*ParamsLinear)),list(FitDF['Error']),ParamsLinear)
print(ChiStats)
```

A3.3 Subtraction.py

This Python script was resopnsible for reading in the data files for the three known sources and the mystery source and subtracting the background that was there in all spectra. This background had to be correctly normalised to the same time period as the sources were taken for, so that way the correct amount of background is subtracted.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import scipy as sc
import pandas as pd

# If statements that control what the code does!
PlotMysterySource = True
PlotKnownSources = True

# Plot controlls
TitleFont = {'size':'23', 'color':'black', 'weight':'bold'}
AxTitleFont = {'size':'20'}

if PlotMysterySource:
  BgWithShield = "WeekendBgWithShield_003_eh_1"
  BgWithShieldDF = pd.read_csv(BgWithShield+ ".dat", sep=r"\s+",names = ['channel number','count number'])

  MysterySource = "MysterySource-24HrRun_001_eh_1"
  MysterySourceDF = pd.read_csv(MysterySource+ ".dat", sep=r"\s+",names = ['channel number','count number'])

  # Normalises the timings of the two runs to 24 hours each
  BgWithShieldDF['count number2'] = BgWithShieldDF['count number']
  BgWithShieldDF['count number'] = BgWithShieldDF['count number'] / ( 239068  ) * 86400

  MysterySourceDF['count number'] = MysterySourceDF['count number'] - BgWithShieldDF['count number']

  # Fit parameters
  a = 2.63977565e-06
  b = 7.04767648
  c = -1.91414134e-01

  # Old fit params for comparison
  aold = 7.012
  bold = 8.170

  MysterySourceDF['Energies'] =  np.sqrt( (MysterySourceDF['channel number'] - c + (b**2.0/ (4 * a)))/a ) - (b/(2*a))
  MysterySourceDF['Energies2'] = (MysterySourceDF['channel number'] -bold)/aold

```

```
41    BgWithShieldDF['Energies'] =  np.sqrt( (BgWithShieldDF['channel number'] - c +
      ↪  (b**2.0/ (4 * a)))/a ) - (b/(2*a))
42
43    # Plotting the mystery source on a log-y axis with nice formatting
44    plt.semilogy(MysterySourceDF['Energies'],MysterySourceDF['count
      ↪  number'],label="Mystery Source")
45    plt.semilogy(BgWithShieldDF['Energies'],BgWithShieldDF['count
      ↪  number2'],label="Background With Shield")
46    plt.xlabel('Gamma Energy [keV]',**AxTitleFont)
47    plt.ylabel('Log-10 of count number',**AxTitleFont)
48    plt.title('Mystery Source With Background Subtracted',**TitleFont)
49    plt.xlim(0,2216)
50    plt.ylim(1)
51    plt.legend()
52    plt.show()
53
54  if PlotKnownSources:
55    BgWithShield = "WeekendBgWithShield_003_eh_1"
56    BgWithShieldDF = pd.read_csv(BgWithShield+ ".dat", sep=r"\s+",names = ['channel
      ↪  number','count number'])
57
58    Sodium = 'Sodium22-24HrData_002_eh_1'
59    SodiumDF = pd.read_csv(Sodium+ ".dat", sep=r"\s+",names = ['channel number','count
      ↪  number'])
60
61    Barium = 'Barium133-24HrRun_006_eh_1'
62    BariumDF = pd.read_csv(Barium+ ".dat", sep=r"\s+",names = ['channel number','count
      ↪  number'])
63
64    Cobalt = 'Cobalt60-24HrData_004_eh_1'
65    CobaltDF = pd.read_csv(Cobalt+ ".dat", sep=r"\s+",names = ['channel number','count
      ↪  number'])
66
67    # Normalises the background to the correct timings
68    BgWithShieldDFCobalt = BgWithShieldDF.copy()
69    BgWithShieldDF['count number'] = BgWithShieldDF['count number'] / ( 239068  ) *
      ↪  86400
70    BgWithShieldDFCobalt['count number'] = BgWithShieldDF['count number'] / ( 239068  )
      ↪  * (86400/12.0)
71
72    # Subtracts the background from the original data
73    SodiumDF['count number'] = SodiumDF['count number'] - BgWithShieldDF['count number']
74    BariumDF['count number'] = BariumDF['count number'] - BgWithShieldDF['count number']
75    CobaltDF['count number'] = CobaltDF['count number'] - BgWithShieldDFCobalt['count
      ↪  number']
76
77    # Fitting parameters
78    a = 2.63977565e-06
79    b = 7.04767648
80    c = -1.91414134e-01
81
82    # Converting channel numbers to energies
83    SodiumDF['Energies'] = np.sqrt( (SodiumDF['channel number'] - c + (b**2.0/ (4 *
      ↪  a)))/a ) - (b/(2*a))
```

```
84  BariumDF['Energies'] = np.sqrt( (BariumDF['channel number'] - c + (b**2.0/ (4 *
    ↪  a)))/a ) - (b/(2*a))
85  CobaltDF['Energies'] = np.sqrt( (CobaltDF['channel number'] - c + (b**2.0/ (4 *
    ↪  a)))/a ) - (b/(2*a))
86
87  # Now plotting the different elements on different log-y plots.
88  plt.figure(1)
89  plt.semilogy(SodiumDF['Energies'],SodiumDF['count number'],label="Sodium 22
    ↪  data",color='yellow')
90  TitleFont = {'size':'25', 'color':'black', 'weight':'bold'}
91  AxTitleFont = {'size':'22'}
92  plt.xlabel('Gamma Energy [keV]',**AxTitleFont)
93  plt.ylabel('Log-10 of count number',**AxTitleFont)
94  plt.title('Sodium 22, no background, energies',**TitleFont)
95  plt.xlim(0)
96  plt.ylim(1)
97  plt.legend()
98
99  plt.figure(2)
100 plt.semilogy(BariumDF['Energies'],BariumDF['count number'],label="Barium 133
    ↪  data",color='green')
101 TitleFont = {'size':'25', 'color':'black', 'weight':'bold'}
102 AxTitleFont = {'size':'22'}
103 plt.xlabel('Gamma Energy [keV]',**AxTitleFont)
104 plt.ylabel('Log-10 of count number',**AxTitleFont)
105 plt.title('Barium 133, no background, energies',**TitleFont)
106 plt.xlim(0)
107 plt.ylim(1)
108 plt.legend()
109
110 plt.figure(3)
111 plt.semilogy(CobaltDF['Energies'],CobaltDF['count number'],label="Cobalt 60
    ↪  data",color='blue')
112 TitleFont = {'size':'25', 'color':'black', 'weight':'bold'}
113 AxTitleFont = {'size':'22'}
114 plt.xlabel('Gamma Energy [keV]',**AxTitleFont)
115 plt.ylabel('Log-10 of count number',**AxTitleFont)
116 plt.title('Cobalt 60, no background, energies',**TitleFont)
117 plt.xlim(0)
118 plt.ylim(1)
119 plt.legend()
120 plt.show()
```

A3.4 Resolution.py

This Python file was responsible for plotting, and fitting, the energy resolution of each peak as a function of that peaks energy.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as sciopt
import seaborn as sns
import pandas as pd
sns.set()
import scipy.stats as scistats

def ResolutionFit(E,a,b):
  return a/np.sqrt(E) + b

FitDF = pd.read_csv('Gamma_Peak_Stats_and_Params.csv', names=['Element', 'Fit type', 'Peak number',  'Peak type', 'Energy (keV)', 'Resolution', 'Min', 'Max', 'Mean', 'A', 'Sigma', 'Error' ,'a', 'b', 'chisq', 'Reduced chisq', 'Probchisq'],usecols=list(range(1,18)))

FitDF = FitDF[FitDF['Fit type'] == 'Linear']
FitDF = FitDF[FitDF['Peak type'] == 'Zoom']

FitDF['Energy (keV)'] = FitDF['Energy (keV)'].astype(float)
FitDF['Resolution'] = FitDF['Resolution'].astype(float).multiply(100)
SodiumDF = FitDF[(FitDF['Energy (keV)'] == 511)]
FitDF = FitDF[~(FitDF['Energy (keV)'] == 511)]

DataBa = FitDF[FitDF['Element'] == 'Barium133']
DataCo = FitDF[FitDF['Element'] == 'Cobalt60']
DataNa = FitDF[FitDF['Element'] == 'Sodium22']

Fit, Errors = sciopt.curve_fit(ResolutionFit,FitDF['Energy (keV)'],FitDF['Resolution'])
print(Fit)

TitleFont = {'size':'20', 'color':'black', 'weight':'bold'}
AxTitleFont = {'size':'18'}

plt.figure(1)
Energies = np.linspace(55,1400,14000)
plt.plot(Energies,ResolutionFit(Energies,*Fit),label="Fit")
plt.errorbar(DataBa['Energy (keV)'],DataBa['Resolution'],fmt="o",color='red',label="Barium-133",markersize=7,yerr=(7.05/DataBa['Energy (keV)']),elinewidth=3,capsize=5,capthick=3,marker='o')
plt.errorbar(DataCo['Energy (keV)'],DataCo['Resolution'],fmt="o",color='blue',label="Cobalt-60",markersize=7,yerr=(7.05/DataCo['Energy (keV)']),elinewidth=3,capsize=5,capthick=3,marker='o')
plt.errorbar(DataNa['Energy (keV)'],DataNa['Resolution'],fmt="o",color='yellow',label="Sodium-22",markersize=7,yerr=(7.05/DataNa['Energy (keV)']),elinewidth=3,capsize=5,capthick=3,marker='o')
```

```
38 plt.errorbar(SodiumDF['Energy (keV)'],SodiumDF['Resolution'],fmt="o",color='yellow',l
   ↪ abel="",markersize=7,yerr=(7.05/SodiumDF['Energy
   ↪ (keV)']),elinewidth=3,capsize=5,capthick=3,marker='o')
39 plt.xlabel('Peak Energy [keV]',**AxTitleFont)
40 plt.ylabel('Energy Resolution [%]',**AxTitleFont)
41 plt.title('Energy Resolution as a Function of Gamma Energy',**TitleFont)
42 plt.legend(fontsize=16,fancybox=True,shadow=False)
43
44 plt.figure(2)
45 plt.errorbar(DataBa['Energy (keV)'],DataBa['Resolution']-ResolutionFit(DataBa['Energy
   ↪ (keV)'],*Fit),fmt="o",color='red',label="Barium-133",markersize=7,yerr=(7.05/Data
   ↪ Ba['Energy
   ↪ (keV)']),elinewidth=3,capsize=5,capthick=3,marker='o')
46 plt.errorbar(DataCo['Energy (keV)'],DataCo['Resolution']-ResolutionFit(DataCo['Energy
   ↪ (keV)'],*Fit),fmt="o",color='blue',label="Cobalt-60",markersize=7,yerr=(7.05/Data
   ↪ Co['Energy
   ↪ (keV)']),elinewidth=3,capsize=5,capthick=3,marker='o')
47 plt.errorbar(DataNa['Energy (keV)'],DataNa['Resolution']-ResolutionFit(DataNa['Energy
   ↪ (keV)'],*Fit),fmt="o",color='yellow',label="Sodium-22",markersize=7,yerr=(7.05/Da
   ↪ taNa['Energy
   ↪ (keV)']),elinewidth=3,capsize=5,capthick=3,marker='o')
48 plt.errorbar(SodiumDF['Energy
   ↪ (keV)'],SodiumDF['Resolution']-ResolutionFit(SodiumDF['Energy (keV)'],*Fit),fmt="
   ↪ o",color='yellow',label="",markersize=7,yerr=(7.05/SodiumDF['Energy
   ↪ (keV)']),elinewidth=3,capsize=5,capthick=3,marker='o')
49 plt.xlabel('Peak Energy [keV]',**AxTitleFont)
50 plt.ylabel('Energy Resolution Residual [%]',**AxTitleFont)
51 plt.title('Energy Resolution Residuals',**TitleFont)
52 plt.legend(fontsize=16,fancybox=True,shadow=False)
53 plt.show()
54
55 def ChiSqFunc(Measured,Fitted,Errors,Params):
56   ChiSquared = 0
57   for i in range(len(Measured)):
58     ChiSquared += ((Measured[i] - Fitted[i])**2.0) / ((Errors[i])**2.0)
59   ReducedChiSq = ChiSquared/(len(Measured)-len(Params))
60   ProbChiSq = (1.0 - scistats.chi2.cdf(ChiSquared,len(Measured)-len(Params)) )  *
   ↪  100.0
61   return ChiSquared, ReducedChiSq, ProbChiSq
62
63
64 MeasuredRes = list(DataBa['Resolution'])+list(DataCo['Resolution']) +
   ↪ list(DataNa['Resolution'])
65 FittedRed = list(ResolutionFit(DataBa['Energy (keV)'],*Fit)) +
   ↪ list(ResolutionFit(DataCo['Energy (keV)'],*Fit)) +
   ↪ list(ResolutionFit(DataNa['Energy (keV)'],*Fit))
66 ErrorRes = list(7.05/DataBa['Energy (keV)']) + list(7.05/DataCo['Energy (keV)']) +
   ↪ list(7.05/DataBa['Energy (keV)'])
67
68 ChiStats = ChiSqFunc(MeasuredRes, FittedRes, ErrorRes,Fit)
69 print(ChiStats)
```