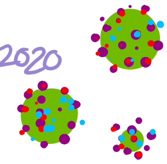
—Measurement of β-ray Spectra—

05/08/2020



Ana Fabela Hinojosa – 27876594

Booking of apparatus: 06/08 from: 10:00 → 16:00 (original)
~~16/08 from 11:00 → 17:00~~ ~~(additional)~~
18/08 from 10:00 → 16:00 (additional)

Aim: we measure the energy spectrum of emitted β-rays and the total energy released by one of the nuclear transitions

Theory: There are 2-types of β-decay, we will focus on:

β^--decay – electrons ejected from a nucleus such that a nuclide transforms into an isobar [1]

The process is described by the equation:

$${}^{A}_{Z}X \rightarrow {}^{A}_{Z+1}Y + e^- + \bar{\nu}$$

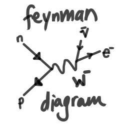
Atom (nuclide)

isobar

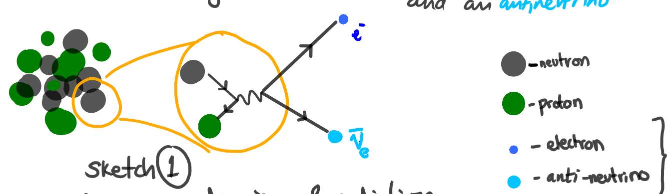
A – atomic number

Z – number of protons

feynman diagram



The β^- process: neutron becomes proton by emitting an electron and an antineutrino



Sketch ①
A neutron undergoing β-radiation
(• reference to particles on the right)
(I ignore W⁻ boson interaction)

The radiated electron may have kinetic energy ranging from zero to nearly the full disintegration energy. [0]
The total energy is divided between nuclide, electron, anti-neutrino [1]

Things to consider • The antineutrino has a rest mass $< 2\times10^{-7} m_e$ [0]

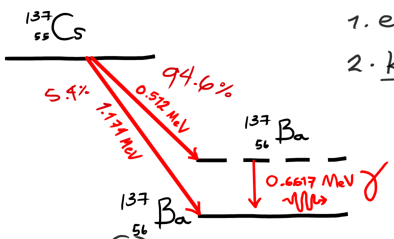
In the lab manual, we are shown in Fig 2. β-ray groups in the decay scheme of ^{137}Cs

HIGHER ENERGY GROUP

An excited state of $^{137}_{56}Ba$ which de-excites by either

1. emitting a γ-ray [E = 661.65 keV]
2. K conversion electron [E = 624.21 keV]

↳ common Internal conversion (IC) electrons come from K shell (1s state) [5]



Sketch ② a copy of: Fig 2 in Lab notes

WHY IS HIGHER ENERGY GROUP DIFFICULT TO DETECT?

is it dependent on the degree of "forbiddenness" of the transition?

nuclide: atomic species [2]

isobar: atoms of different elements that have the same number of nucleons [3]

Conversion electrons

05/08/2020

The nucleus of an atom can be in an excited state after undergoing radioactive decay.

An atom can de-excite if this excess energy is transferred to an orbital electron. (Likely a K-shell electron)

When the electrons are ejected from their orbits by gaining energy they are called conversion electrons and are emitted with a characteristic ENERGY.

The energy of conversion electrons ≈ { The excited state energy of the nucleus minus the binding energy of this inner shell electron to the atom.

[0] momentum spectrum of β particles

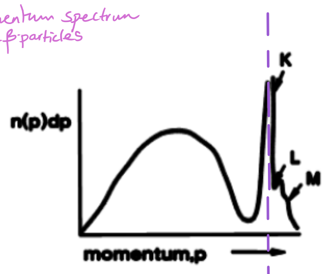


Figure 1: A representative momentum spectrum of beta particles from the decay of ^{137}Cs. The K, L and M conversion electron peaks are also depicted on the right. [0]

K conversion line can be used for calibration of the momentum range of the spectrometer.

We are told in the lab script that:

The ratio of the peak's amplitudes follows $\frac{K}{L+M} = 4$

Therefore we can assume that we will only observe the K conversion line.

— The Spectrometer —

05/08/2020 & 11/08/2020

Our apparatus is a thin magnetic lens spectrometer (Detector: G-M tube with thin-mic. end window)

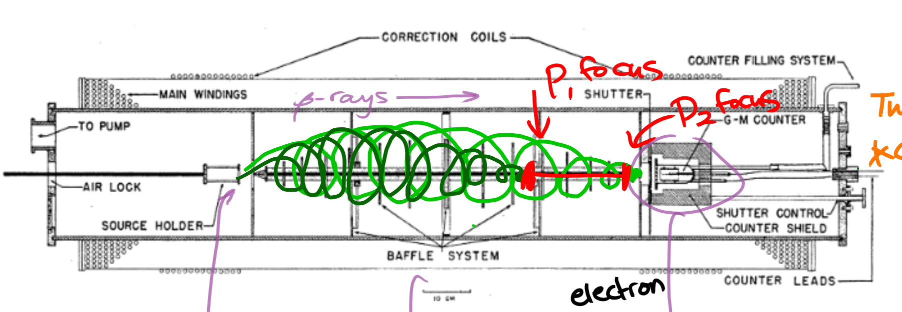


FIG. 4. Solenoidal beta-ray spectrometer.

^{137}Cs

detector

adjusts focus

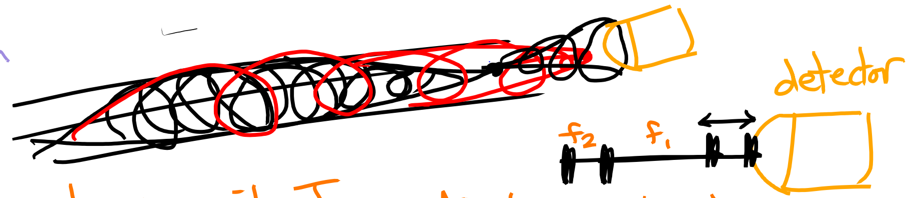
only e⁻ of particular p (ie. T) range are focused onto the detector for a given |B| depends on I of electromagnet

The effect of the magnetic field ⋆a cone of electron trajectories diverging around the source around the instrument's axis and converging (depending on their momentum) [6] on the detector

Focal length

e⁻ of given momentum

3-axis Magnetometer measures $\vec{B}$ in each direction (position is adjustable)

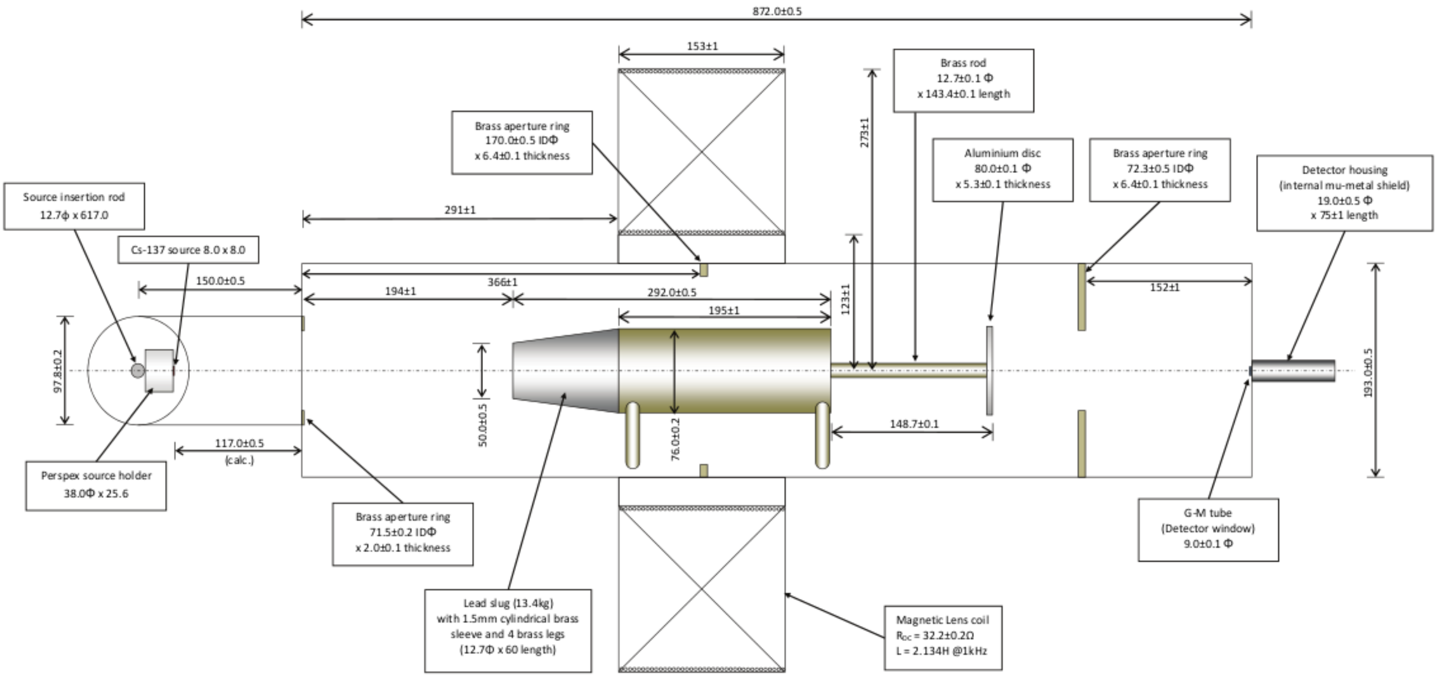


Lens coil I: adjusts focal length

Schematic diagram of our Spectrometer -

05/08/2020

Beta-Ray Spectrometer Schematic Diagram
(Side view)



focus adjustment
Via magnetic field

0 40 80 120 160 200 mm

Scale 1:4
All measurements in millimetres unless specified otherwise

Monash University - 3rd Year Physics Laboratory	
Beta-ray Spectrometer Schematic Diagram (Side View)	
Drawn by: John Golja	**Date:** 31/01/2012

14/08/2020

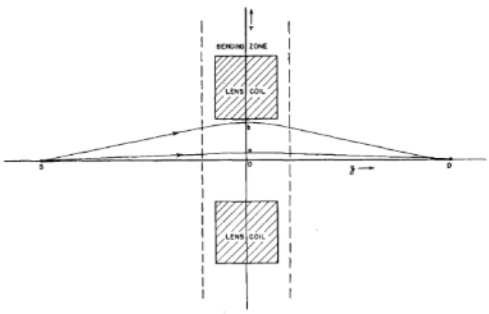


Fig. 1. Section through a typical magnetic lens type beta-ray spectrometer with central baffle emitted, showing source S, detector D, and two rays. The lens current is chosen to focus the ray which is closer to the axis.

*** Tracerlab, Inc., Boston, Massachusetts.

[10]

[11] The effect of magnetic fields on the motion of charged particles

[10] $P = kI \underset{?}{=} R = B\rho$ is the rigidity

determined by spectrometers geometry.

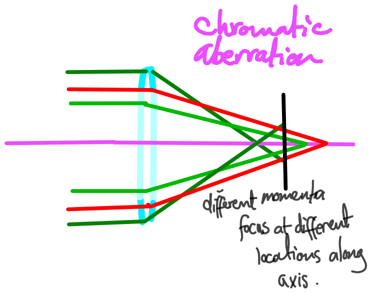
— Script Notes — 05/08/2020

The effect of the magnetic ($\vec{B}$) field:

- the focal length of the lens depends on the electron momentum

for a given $\vec{B}$:

- electrons with specific momentum are focused on the detector
- electrons carrying other momenta will focus on different points (Chromatic aberration) and will removed
- γ-rays are prevented from reaching the detector by a lead block on the axis



By varying the $\vec{B}$ field we focus the detector to various momenta

Resolution ~ change size and shape of baffles

typical resolution is ~2-3 %

*We need to divide the number of counts at each magnetic field $\vec{B}$ (reading $\vec{B}$)

Mathematical model

strength magnetic field → coil current ↓

in the absence of ferromagnetic materials: $B \propto I$

- Momentum (p)

$p \propto B$

Energy kinetic ? yep

β-decay depends on Energy of transition (T)

part of T goes into "creating" the particle

(Contrast α & γ decay)

we must work with Total-mass energy (ω) of the β-particle.

(1) $W^2 = p^2c^2 + \underbrace{m}_{e^-\text{ rest mass}}{}^2c^4$ → relativistic units

(2) $W = T + mc^2$ (since m is rest mass)

Questions

05/08/2020

(1) $W^2 = p^2c^2 + m_0^2c^4$

(2) $W = T + m_0c^2$

1. By converting into relativistic units, where momentum $P = p/m_0c$ and energy $W = w/m_0c^2$, derive $W^2 = 1 + P^2$ from Equation (1).

$P = p/m_0c$ (relativistic momentum)

$W = w/m_0c^2$ (relativistic energy)

derive $W^2 = 1 + P^2$

we know (1) $w^2 = p^2c^2 + m_0^2c^4$

where $P = p/m_0c$

$\therefore P m_0 c = p \Rightarrow (P m_0 c)^2 = p^2$

Then (1) becomes:

$w^2 = (P m_0 c)^2 c^2 + m_0^2 c^4$

$\therefore w^2 = P^2 m_0^2 c^4 + m_0^2 c^4$

$\therefore w^2 = (P^2 + 1) m_0^2 c^4$

since $W^2 = \dfrac{w^2}{m_0^2 c^4} \Leftrightarrow W^2 = P^2 + 1$ □

later on we refer to w as

$\overline{W} = \sqrt{P^2 + 1}$

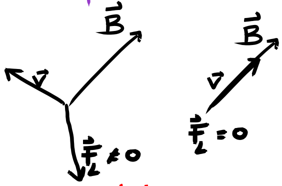
2.

What is the effect of the earth's magnetic field on the path of electrons in the spectrometer and how may this effect be nullified? Note the orientation of the spectrometer's principal axis relative to the horizontal component of the earth's field. Note: if doing the experiment online, the axis of the spectrometer is oriented (approximately) north-south. Consider how you might use the auxiliary Helmholtz coils and a 3-axis magnetic field probe (which is supplied with the experiment) in your experiment to mitigate these effects.

The magnetic field of earth will alter the trajectory of the electrons by means of the Lorentz force

Lorentz force

$\vec{F}_{total} = \cancel{e\vec{E}} + \underbrace{e\vec{v} \times \vec{B}_{total}}$



× we neglect the electric part

effect of Magnetic field on electrons moving at $\vec{v}$

$e\vec{v} \times \vec{B} = e\vec{v} \times (\vec{B}_{spect} + \vec{B}_{earth})$

40 CLIFFORD M. WITCHER

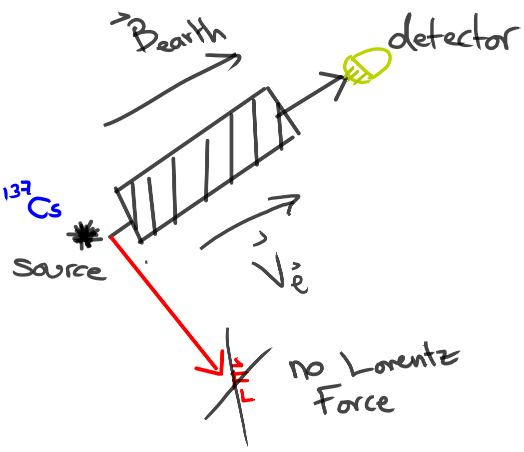
It should be noted that in such an instrument the effect of the earth's field (or other stray fields) cannot be completely neglected in the very low energy region. The axis of the instrument has been placed in the direction of the horizontal component of the earth's field to minimize any effect due to this component. There is no evidence that the operation of the instrument is appreciably affected by the vertical component of the earth's field at the lowest energies so far used (100 kev.) However, a compensating coil for the vertical component is being installed for work down to 10 to 25 kev.

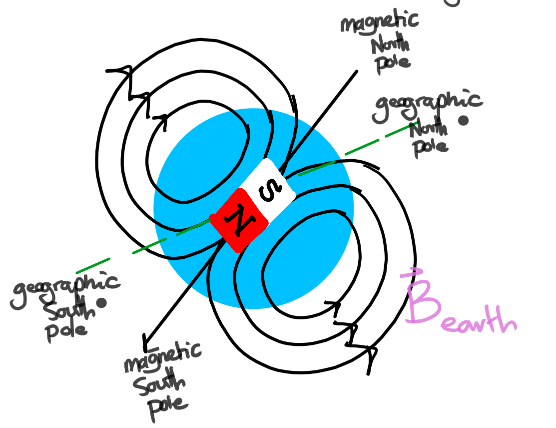
Figure 8 shows t
source after corr
absorption. The i
tistical error of 2
form of the resolu
one can accurately
of any beta-spectr
approximations, p
reasonably strong.
the Ra E spectrum
effect of the resolut
$H\rho$ 5100 (1.11 Me
energy of 3.18 in
on the basis of t

To minimize the undesired effects on the electron beam we align the $\vec{v}$ and $\vec{B}$ vectors

Spectrometer's axis ~ (geographic North Pole • = magnetic North Pole •)

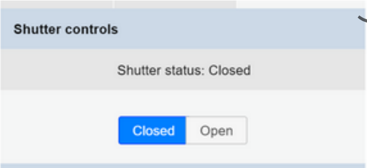
given scale





05/08/2020

3. How might you discover whether there is a constant background counting rate from the detector? Check whether there is such a background and correct for it in analysing your results.



15/08/2020

This took me ages, I know

* Set the shutter to CLOSED
This will shield the radioactive source and so, any detected events will be from background radiation

(Another way could be to de-focus the beam of electrons entirely)

4. You can acquire data in either constant time (time is your independent variable and the number of counts is the dependent variable) or constant count (the number of counts is the independent variable and the time is the dependent variable). You should recall from previous experiments that for a fixed time, the best estimation of the uncertainty in the number of counts, N, is $\sqrt{N}$. If you acquire to a fixed number of counts, what is the uncertainty in the measured time (due to the statistical nature of nuclear decay)?

Data collection

Options

1. counts N vs. Constant time intervals

2. Counts N vs time

Note: for fixed time uncertainty in N is $\sqrt{N}$

What is the uncertainty in time due to statistical nature of nuclear decay?

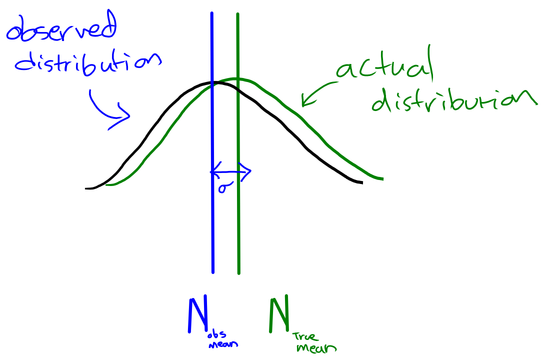
fractional uncertainty in counts: $\frac{\sqrt{N}}{N} = \frac{1}{\sqrt{N}}$

fractional uncertainty in time: $\Delta t = t \cdot \frac{\sqrt{N}}{N}$ $\therefore$ fractional uncertainty is preserved.

5. Consider how you should take your data. From your knowledge of the statistics of nuclear counting (as discussed in Question 4 above), is it more sensible to count for constant times at each current setting in the spectrometer coils or to collect a roughly constant number of counts? Provide a detailed answer on the choice(s) you have decided to make and explain *why* you made them.

Radioactivity is a poissonian process
I will run the experiment for a fixed amount of time and record N
This N will be my observed approximation to the true N_{mean} for this specific process

The observed distribution should have N_{mean} within one standard deviation of the true mean.



Hint: You may want to consider which data points of the momentum spectrum contribute the most to determining the end point energy (see the analysis section on determining the end point energy from the Kurie plot). How do you minimise the fractional uncertainty of these points? You may also want to consider how to minimise the fractional uncertainty for each data point.

05/08/2020

The experimental set-up

Apparatus

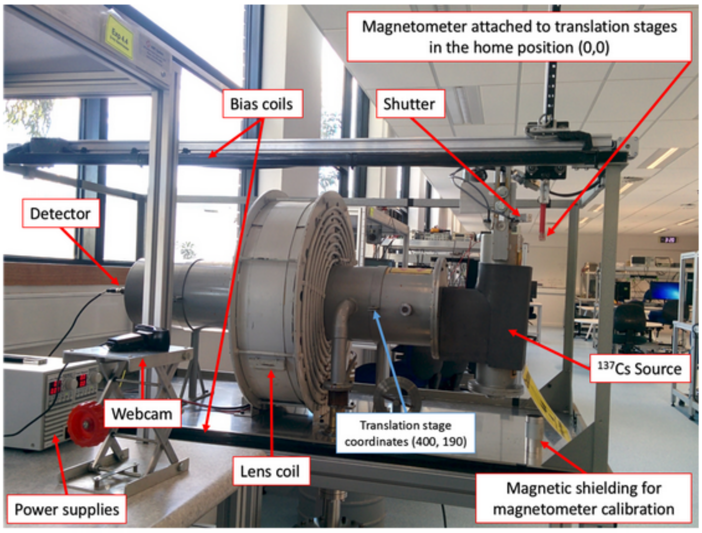
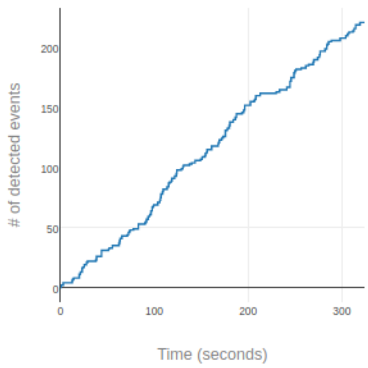


Figure 3: A labelled diagram of the beta-ray spectrometer apparatus located in the laboratory.

Control interface of the apparatus

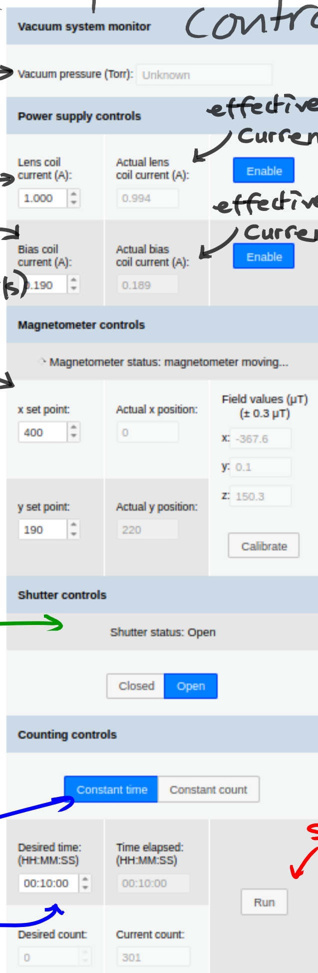
open shutter

magnetometer (Home: (0,0))



count rate plot

* Experimental controls



Not active

Lens coil (I) adjustable

effective current

focus (Helmholtz coils)

effective current for focus

Recommended (400, 190) for measuring $\vec{B}$ close to β-ray path

visible in webcam view.

experimenter choice for analysis

manually set

start experiment.

Experimental Procedure

06/08/2020

1 RUN

To familiarise myself with the control system

1. I decided to make half integer increments to the lens coil current *

   Starting with: lens coil current to 1.000A (0.9932±0.0005 A)

   I pressed $\boxed{\text{enable}}$

2. I will not enable the bias coil current

3. I will not adjust the magnetometer position

4. I set the counting controls as $\boxed{\text{constant time}}$

   and set the desired running time as 10 minutes (~600.000±0.015s)

I repeated these steps 6 times

Observing the data I see that when I have used half integer steps (ie. Lens coil current = 1.500 A 2.500A 3.500A)

The Data table does not record my Lens coil current * ← is this a control system Bug ??

- The bias coil current will be adjusted in my next set of experimental runs

- The next obvious problem is the lack of magnetometer calibration this is to be corrected in my next set of experimental runs

TODO:

0. what is happening with my set up, why do I not get a current reading? sometimes?
1. Should make sure magnetometer is in optimal position
2. NEED to figure out what the 3-axis magnetic field probe is doing
3. * Eventually obtain full momentum spectra by recording # counts in a time period for various lens coil currents

   figure out what size increments are adequate to resolve momentum spectrum

4. figure out if there is Background counting rate (?)

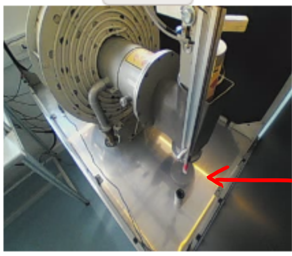
done!
15/08/2020

Experimental Procedure

06/08/2020

Making sure magnetometer is in optimal position *

For the next run I adjusted the coordinates of the magnetometer (0,0) → (400,190)



This is a screenshot of the live camera after the adjustment.

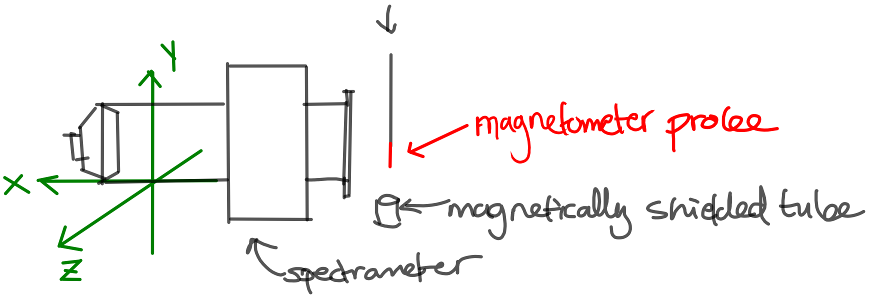
magnetometer probe (400, 190)

Calibrating the magnetic field probe

The magnetic field probe readings will initially be offset from the actual present field. You will likely see a significant "z" component, despite Earth's field not containing a significant component in this direction.

Note that the axes of the magnetometer probe correspond as follows:

- x: The component along the principle axis of the spectrometer (the path between the source and the detector.
- y: The vertical component with respect to the ground (and orthogonal to x).
- z: The horizontal component orthogonal to x.



when the probe is shielded we consider the displayed values as an accurate measurement of the present magnetic field within 0.3 μT

- I proceeded to calibrate the magnetometer probe.

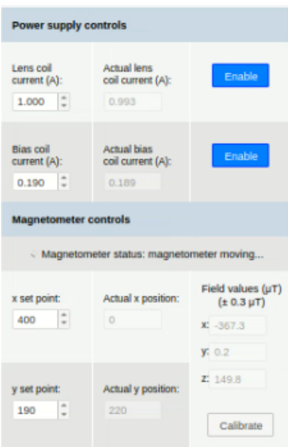
2 RUN (Figuring out how to minimize the effect of the earth's $\vec{B}$ field)

After adjusting the magnetometer probe

1. I returned the Lens coil current to the initial value 1.000 A (0.9934 ± 0.0005 A)

After increasing the bias coil current to 0.190 A (0.1890 ± 0.0005 A) I noticed that this value makes the y-component value of the Field values registered by the 3-axis magnetometer very close to zero.

I cannot adjust the bias coil current in a way that decreases the other components.



Therefore I am going to try this as my first attempt to cancel the magnetic field of the background

I press $\boxed{\text{run}}$

The experiment runs for 10 minutes (600.001 seconds)

Experimental Procedure

06/08/2020

(2.) for the next coil current value:

I want to find out what is going on with my measurements when I do half integer steps!

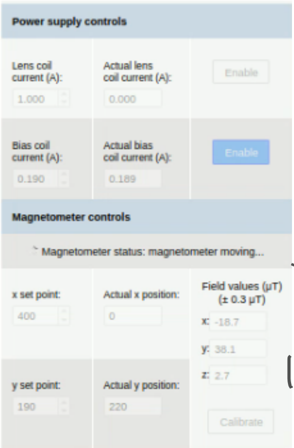
I won't increase the lens current (yet)

The process is similar to (1) RUN

1. Lens coil current = 1.000 A (still)
2. press $\boxed{\text{enable}}$

another difference is → 3. The bias coil current is set to 0.190 A (0.1890 ± 0.0005 A)

4. experiment is Not registering a Lens coil current



After observing the obtained data, I think I know what happened

— I didn't understand the control system interface —

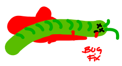
In (1) RUN every time I changed the lens current I pressed $\boxed{\text{enable}}$.

In (2) RUN I didn't change the lens current but I pressed $\boxed{\text{enable}}$.

In the assumption that I was "activating" the experiment

I suspect that I disabled the lens current every second time I pressed $\boxed{\text{enable}}$

This possibly wasted ~ 90 min of data adquisition time.



Bug fix

The bug was inside me all along!

*What did I learn from this?

1. How the buttons work! Brilliant!

* Since we can turn of the lens current completely we can use our magnetometer probe to read earth's $\vec{B}$ field and adjust the bias coil current until we cancel earth's $\vec{B}$ field out completely.

(seems obvious in Hindsight)

Experimental Procedure

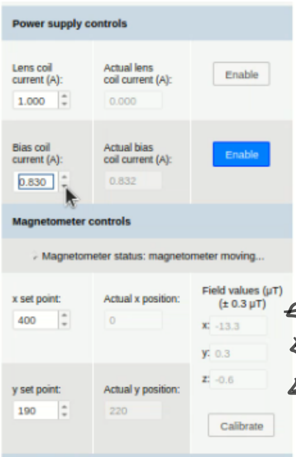
06/08/2020

③ RUN

Once again Starting with:

① lens coil current to 1.000A (0.9934 ±0.0005 A)

I pressed enable ← To disable the lens current briefly



this is earth's B field minimized

② I increased the bias coil current until I see all displayed values as close to zero as I can get them.

∴ set value for bias coil current is: 0.830A (This # varies slightly every run)

- Given all the time "wasted", in order to get enough data I decided to make quarter integer increments to the lens coil current (Hope this optimizes for both, number of data points & resolution of the spectrum)

* ③ Then the process is as follows: *

- Increase the lens coil current by 0.250 A
- Leave bias coil current as is, but verify that 3-axis are still close to zero every run. By briefly disabling the lens coil current.
- press run to detect N for 10 minutes
- save plot "counts per time", record corresponding time
- Repeat until reaching lens coil current = 3.500 A

TODO:

~~0. what is happening with my set up, why do I not get a current reading? sometimes?~~

~~1. Should make sure magnetometer is in optimal position~~

~~2. NEED to figure out what the 3-axis magnetic field probe is doing~~

* 3. Eventually obtain full momentum spectra by recording # counts in a time period for various lens coil currents

figure out what size increments are adequate to resolve momentum spectrum ✓

4. figure out if there is Background counting rate

* A possible improvement to the Laboratory script:

The "experiment Procedure" tab should be moved to be before "Reading the csv file" tab. This might just be my fault but I didn't understand how important the calibration was just from the theory, it took me some time to learn this ⇒ wasted time!

Analysis

06/08/2020

3 RUN (I will only consider this data in my analysis)

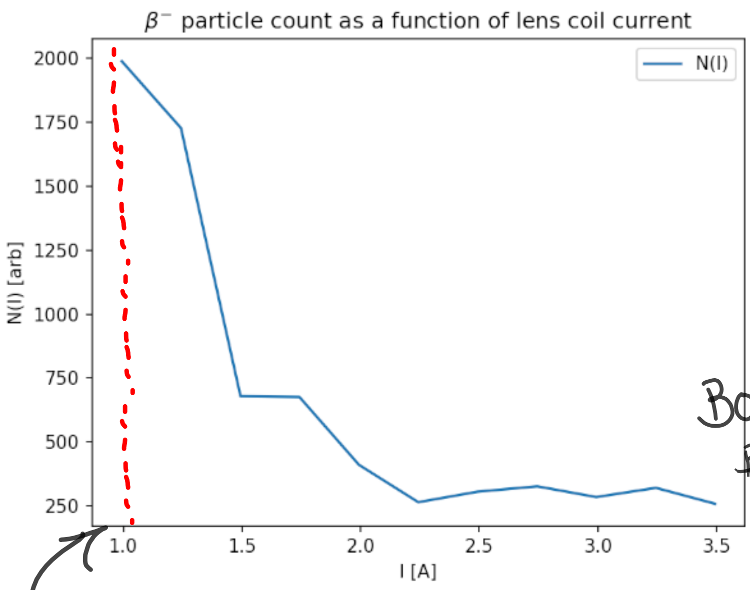
Valid Data

•	Duration (s)	Count	Lens coil current (A)	u(Lens coil current) (A)	Bias coil current (A)	u(Bias coil current) (A)
1	600.005	1988	0.99343	0.0005	0.83093	0.0005
2	600.005	1727	1.24269	0.0005	0.83106	0.0005
3	600.008	679	1.49371	0.0005	0.83091	0.0005
4	600.007	676	1.74207	0.0005	0.83111	0.0005
5	600.004	411	1.99288	0.0005	0.83104	0.0005
6	600.004	265	2.24291	0.0005	0.83106	0.0005
7	600.005	306	2.49286	0.0005	0.83096	0.0005
8	600	327	2.74431	0.0005	0.83099	0.0005
9	600.009	285	2.99308	0.0005	0.83097	0.0005
10	600.002	321	3.24456	0.0005	0.83102	0.0005
11	600.006	259	3.49335	0.0005	0.831	0.0005

•	Magnetometer x position	Magnetometer y position	Magnetometer x-axis field	u(Magnetometer	Magnetometer y-axis field	u(Magnetometer	Magnetometer z-axis field strength (μT)	u(Magnetometer
1	400	190	-362.61856	0.3	-35.86697	0.3	148.2517	0.3
2	400	190	-449.99378	0.3	-44.91145	0.3	185.44758	0.3
3	400	190	-537.9354	0.3	-53.9059	0.3	223.45595	0.3
4	400	190	-625.54941	0.3	-62.67791	0.3	261.6502	0.3
5	400	190	-714.12209	0.3	-71.35809	0.3	300.39517	0.3
6	400	190	-803.34455	0.3	-80.01129	0.3	338.86251	0.3
7	400	190	-891.95172	0.3	-88.22483	0.3	377.46732	0.3
8	400	190	-981.25639	0.3	-96.53003	0.3	416.37661	0.3
9	400	190	-1069.90218	0.3	-104.74846	0.3	454.82582	0.3
10	400	190	-1159.91273	0.3	-112.64221	0.3	494.20548	0.3
11	400	190	-1249.45357	0.3	-120.52033	0.3	532.30743	0.3

N(I) vs I_{lens}

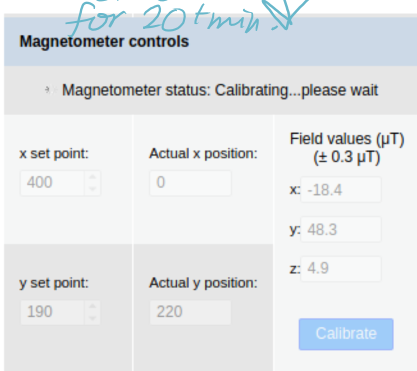
14/08/2020



I don't think that I have enough current values,
+ Starting at 1A was probably a bad choice!

Booked another lab session ~~16/08/2020~~
Bad Luck!

Stayed stuck for 20 min



Booked another lab session 18/08/2020


Bugs Bugs Bugs

Updating my understanding

14/08/2020

Theory (Round II)

[9] Siegbahn

An electron (charge e^-, velocity $\vec{v}$) moving in a homogeneous magnetic field $\vec{B}$ in a plane perpendicular to the lines of Force, has radius of curvature ρ, the kinetic energy is:

$$B e \vec{v} = \frac{mv^2}{\rho} = T$$

where $m = \dfrac{m_0}{\sqrt{1-(v/c)^2}}$ (m_0 is rest mass)

$\therefore$ the momentum is expressed as a $B\rho$-value

$$p = mv = eB\rho$$

This relation is the reason why we use momentum coordinates as opposed to Energy

Since $E_{kinetic} = \frac{p^2}{2m}$ plotting the distribution as an energy distribution $n(E)dE$

Then the measures relate as $dE = \frac{p}{m}dp$.

The corresponding <u>intensities</u> in the spectrum plot ($n(E)$ vs E) are multiplied by $\frac{m}{p}$.

The momentum distribution of the electrons in a magnetic Spectrometer is $N(p)\,dp$

For this experiment, we use a spectrometer of fixed geometry * and variable $\vec{B}$

$\therefore$ <u>The resolution</u> (uncertainty): $R = \dfrac{\Delta(B\rho)}{B\rho} = \dfrac{\Delta p}{p} \sim 2-3\%$

where Δp is a measure of The <u>accepted</u> momentum band

– The rigidity –

[11] The effect of magnetic fields on the motion of charged particles $\rightarrow$ $R = B\rho = \dfrac{p}{e}$ where
- B is magnetic field
- ρ is gyroradius of the particle due to field
- p is momentum
- e is charge

A measure of the momentum of the particle

Higher momentum $\Rightarrow$ higher "resistance" to deflection by a magnetic field.

Quade, Halliday (1947)

[10] The rigidity: $p = kI$

Where k is a constant determined by the spectrometer's geometry *

The current relates to the magnetic field as

$$I = \frac{e\rho B}{k}$$

where:
- B is magnetic field
- ρ is gyroradius of electron due to B
- e is charge of electron

Theory

14/08/2020

[8] Relativistic momentum and Energy for β^- decay

In Relativistic units

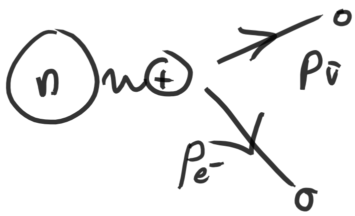
- electron's energy -

$$\bar{\omega} = \bar{T} + 1 = \sqrt{\bar{p}^2+1}$$

(in SI: $\omega = T + mc^2 = \sqrt{p^2c^2 + m_0c^2}$)

The rest energy of an electron is 0.511 MeV

∴ Decay energy : $\omega_0 \geq 0.511\,\text{MeV}$



The antineutrino can't be detected: $E_\nu = cp_\nu$

The electron has kinetic energy $T = \bar{\omega} - 1 = \sqrt{\bar{p}^2+1} - 1$

Since the isobar is very massive compared to the other particles

[Total decay energy: ∴ $\omega_0 \simeq T + E_\nu$] ⇐ momentum is conserved

— The number of electrons with a particular momentum —

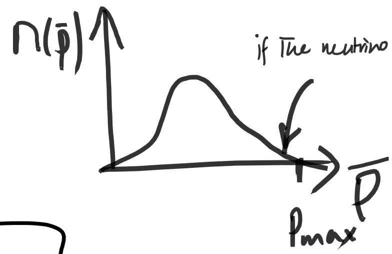
ω_0 is energy of transition

constant = 55

$$n(\bar{p}) = K_1 F(Z, \omega)\, \bar{p}^2 (\omega_0 - \omega)^2 S_n(\omega)$$

Notice that:

* $n(\bar{p}) = 0 \begin{cases} \text{if } \bar{p} = 0 \\ \text{elif } \bar{p} = p_{max} \text{ if } \bar{p}_{max} \neq 0: (\omega_0 - \omega)^2 = 0 \text{ Then } \omega_0 = \omega \end{cases}$



if The neutrino carries no momentum/energy ∴ $E_\nu = 0$

[we want to measure $\bar{p}_{max}$ to calculate $T_{max} = \sqrt{\bar{p}_{max}^2+1} - 1$ which will let us determine ω_0]

Theory Analysis

14/08/2020

–Calibration of energy and momentum range–

[0] The theories of beta-decay depend on the energy of the transition, but because part of the energy of the transition has to go into creating the beta particle (contrast alpha and gamma decay), we must work with the total mass-energy, w, rather than kinetic energy, T, of the beta. Recall that:

$$w^2 = p^2c^2 + m_0^2c^4\,, \tag{1}$$

where p is the momentum (of the beta particle), c is the speed of light, m_0c^2 is the rest mass-energy of the electron and

$$w = T + m_0c^2\,. \tag{2}$$

we are told that $T_k = 624.21$ keV

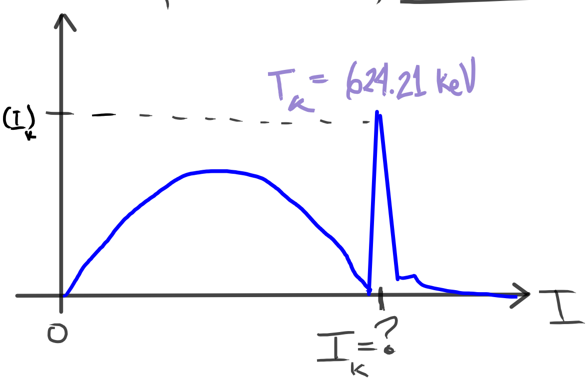
(is the conversion energy spike?)

we know $\bar{T} = \bar{\omega} - 1 = \sqrt{\bar{p}^2+1} - 1$ is the kinetic energy

$$\Rightarrow \bar{T}_k = \sqrt{\bar{p}_k^2+1} - 1$$

$$\therefore \bar{P}_k = \sqrt{(\bar{T}_k+1)^2 - 1}$$

n as function of current



Then our first plot. n(I) should look like ⟶

momentum is expressed as:

* $p = eB\rho$ [9]

* $p = kI$ [10]

Then $\boxed{P_k = KI_k}$

To find the constant K we use $\sqrt{(\bar{T}_k+1)^2 - 1} = KI_k$

$$\therefore \frac{\sqrt{(\bar{T}_k+1)^2 - 1}}{I_k^2} = K$$ ⟵ WOAH!

after we have found our constant of proportionality (K) we can verify

The momentum Spectruem

[0]

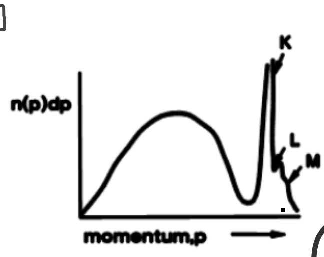


Figure 1: A representative momentum spectrum of beta particles from the decay of ^{137}Cs. The K, L and M conversion electron peaks are also depicted on the right.

$kI = p$

Theory / Analysis

14/08/2020

How Long do we want to count events for?

To determine the time interval used to count events, we need to use our uncertainty

$$\frac{u(n)}{n} = \frac{\sqrt{n}}{n} = \frac{1}{\sqrt{n}}$$

if we want our uncertainty to be

$$u(n) \leq 5\%$$

Then

$$\frac{1}{\sqrt{n}} \leq \frac{1}{20} \therefore n = 20^2 = 400$$

first estimate for time interval if our first attempt at the experiment had $t_1 \approx 600\,s$

Then $\frac{n}{t} = \frac{2000}{t_1} = \frac{400}{t_2} \Rightarrow \frac{10}{3} = \frac{2}{t_2} \Rightarrow t_2 = 360\,s$

in our first attempt at the experiment we observed that the count decreased as the lens current increased. so this should be in our mind as we run the experiment in order for our count to continue having an uncertainty of $\leq 5\%$.

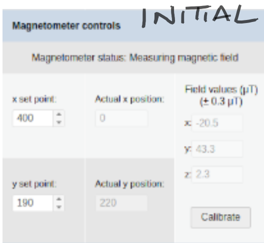
Experimental Procedure (Round II)

18/08/2020

Calibrate magnetic field probe

* the coordinates of the magnetometer are set to: (400, 190)

INITIAL



These values → are an accurate measurement of the field within $0.3\mu T$

Keep in mind:

1. what size increments are adequate to resolve momentum spectrum
- Initial increments 0.1A
- Run at 11:01am was set to incorrect current 0.0034A

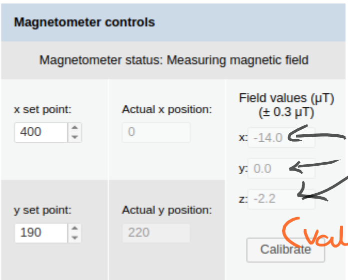
2. ~~do we need to modify the time interval?~~

Nulling the Earth's field

lens coil current to 0A (0 ±0.0005A)

press $\boxed{\text{enable}}$ ← To disable the lens current briefly

AFTER earth's $\vec{B}$ field is "minimized"



These values oscillate (value ± 0.03μT)

$\boxed{\text{ENABLE}}$ bias coil current and increase the bias coil current until all displayed field values are close to zero

bias coil current is: 0.720 A (0.716 ±0.0005A)

Background rate count.

- Set counting controls to $\boxed{\text{constant time}}$
  - set the count running time to 360s (± 0.015 s) ← eyeball estimate
- Set shutter to $\boxed{\text{closed}}$
- proceed to count the background radiation for 4 runs (360s each run), record each count run.

Background counts
✓ 188
✓ 158
✓ 147
✓ 192

(1) β⁻ particle count RUN

– Calibration of energy and momentum range –

The process is as follows:

1. Set the shutter status to $\boxed{\text{open}}$
2. $\boxed{\text{ENABLE}}$ lens coil current.

we start with lens coil current set to 0A (0.0000 ±0.0005A)

** lens coil current increments will be 0.1A (0.0940 ±0.0005A) **

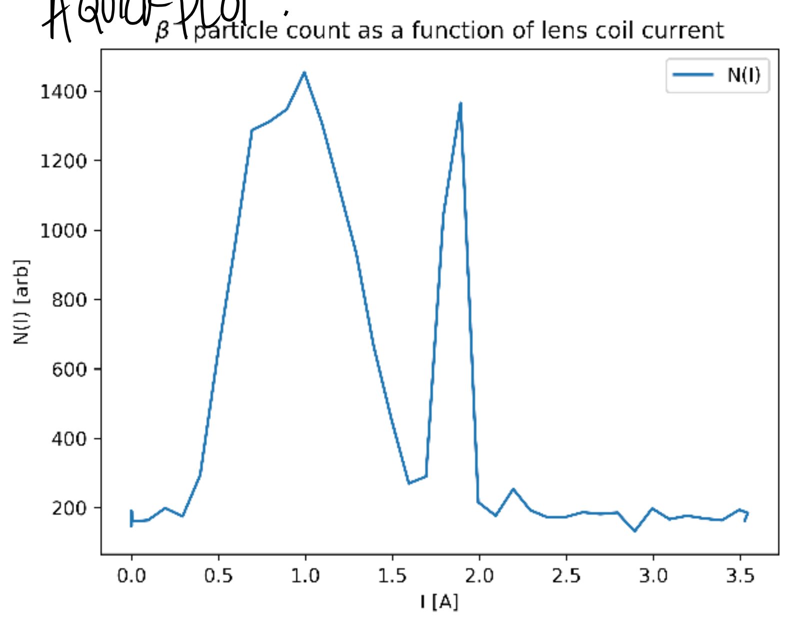
* Leave bias coil current as is, but verify that 3-axis are still close to zero every run. By briefly disabling the lens coil current prior to running count. *

- press $\boxed{\text{run}}$ to detect current dependent count ($n(I)$)
- increase current by 0.1A
- Repeat until reaching MAX lens coil current = 3.6000A (3.5245 ±0.0005A)

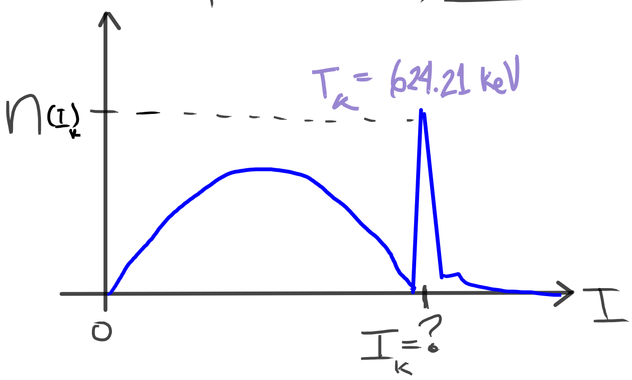
Analysis (Round II)

18/08/2020

A Quick Plot :

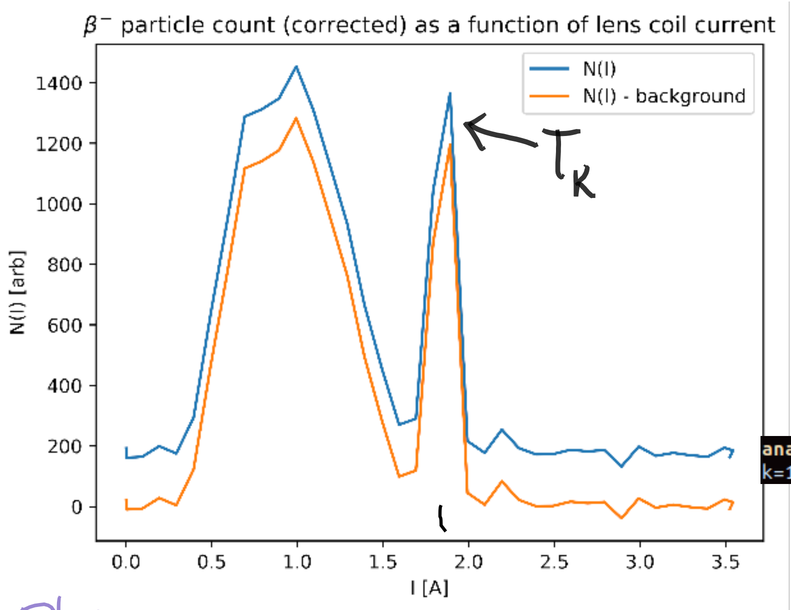


n as function of current



Since $T_K = 624.21$ keV

$I_K = 1.8916 \pm 0.0005$



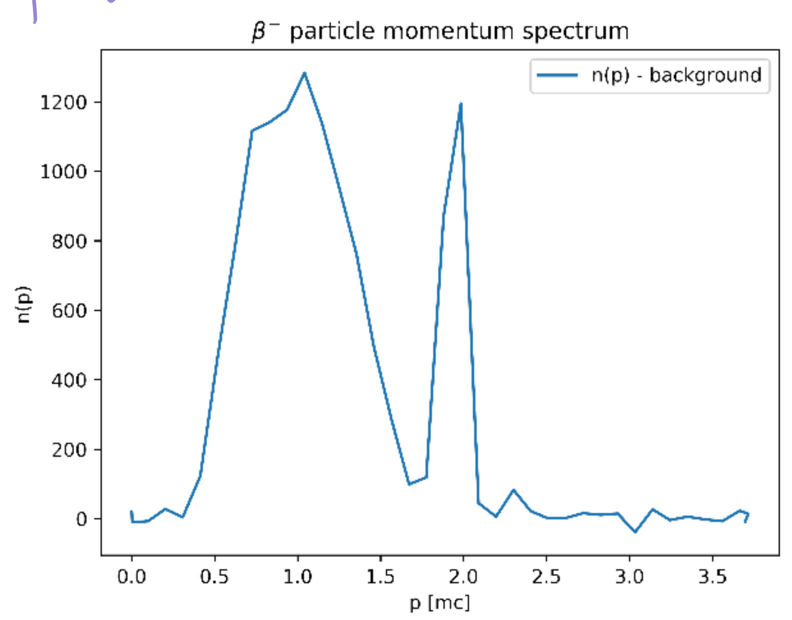
and we know $\bar{p}_K = k I_K = \sqrt{(\bar{T}_K + 1)^2 - 1}$

$$\therefore \frac{\sqrt{(\bar{T}_K + 1)^2 - 1}}{I_K^2} = k$$

```
# Finding constant of proportionality in p = kI
#calibration peak (K) index of k peak is i=20
T_K = 624.21 * keV / rel_energy_unit
k = np.sqrt((T_K + 1)**2 - 1) / lens_current[20]
print(f"{k=}")
```

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 betarays.py
k=1.048711908196117
```

Plot : The momentum Spectrum



$\bar{p} = kI$

```
# The momentum spectrum
lens_current = np.array(lens_current)
p_rel = k * lens_current
# print(p_rel)

plt.plot(
        p_rel, count - avg_background_count, marker="None",
        linestyle="-",
        label="n(p) - background"
)
plt.title(r"$\beta^{-}$ particle momentum spectrum")
plt.xlabel("p [mc]")
plt.ylabel("n(p)")
plt.legend()
spa.savefig('count_vs_momentum_no_background.png')
plt.show()
```

what about uncertainty ?

Analysis (Round II)

18/08/2020

— Uncertainty in $\boxed{p = kI}$

we calculated $p_k = k I_k$

where $k = \dfrac{\sqrt{(T_k + 1)^2 - 1}}{I_k^2}$

```
# Finding constant of proportionality in p = kI
#calibration peak (K) index of k peak is i=20
T_K = 624.21 * keV / rel_energy_unit
k = np.sqrt((T_K + 1)**2 - 1) / lens_current[20]
# print(f"{k=}")
u_k = k * (0.0005 / lens_current[20])
print(f"absolute uncertainty: {u_k = }")
print(f"fractional uncertainty: {(u_k / k) = }")
```

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 betarays.py
absolute uncertainty: u_k = 0.00027720094385203994
fractional uncertainty: (u_k / k) = 0.00026432516088126777
```

① The uncertainty in k:

$$u(k) = k\sqrt{\left(\frac{u(T_k)}{T_k}\right)^2 + \left(\frac{u(I_k)}{I_k}\right)^2}$$

where $u(T_k) = 0$ and $u(I_k) = 0.0005\,A$

$$\therefore u(k) = k\sqrt{\left(\frac{u(I_k)}{I_k}\right)^2} = k\,\frac{0.0005\,A}{1.8916\,A} = 0.0002$$

② The uncertainty in p:

$$u(p) = p\sqrt{\left(\frac{u(k)}{k}\right)^2 + \left(\frac{u(I)}{I}\right)^2}$$

```
# The momentum spectrum
lens_current = np.array(lens_current)
p_rel = k * lens_current
u_p_rel = p_rel * np.sqrt((u_k / k)**2 + (0.0005 / lens_current)**2)
print(f"absolute uncertainty u(p_rel):\n {u_p_rel}")
print(f"fractional uncertainty u(p_rel) / p_rel:\n {(u_p_rel / p_rel)}")
```

```
absolute uncertainty u(p_rel):
 [0.00052436 0.00052436 0.000525   0.00052709 0.00053061 0.00053563
 0.00054192 0.00054948 0.0005584  0.00056843 0.00057983 0.00059226
 0.0006057  0.00061988 0.000635   0.00065088 0.00066813 0.00068566
 0.0007036  0.00072235 0.00074155 0.00076167 0.00078212 0.00080299
 0.00082383 0.00084554 0.00086743 0.00088994 0.00091269 0.00093492
 0.00095797 0.00098148 0.00100485 0.00102899 0.00105294 0.00107704
 0.00110124 0.00111333 0.00110882]
fractional uncertainty u(p_rel) / p_rel:
 [2.78636364e+01 1.46833251e-01 5.32672848e-03 2.59748527e-03
 1.72679689e-03 1.29521414e-03 1.04668075e-03 8.84174670e-04
 7.68757889e-04 6.84655465e-04 6.19267215e-04 5.68515632e-04
 5.28049702e-04 4.95603205e-04 4.68633808e-04 4.46161339e-04
 4.26516001e-04 4.10232132e-04 3.96417943e-04 3.84305110e-04
 3.73812227e-04 3.64435428e-04 3.56244631e-04 3.49012222e-04
 3.42707174e-04 3.36939560e-04 3.31813574e-04 3.27140414e-04
 3.22941074e-04 3.19266567e-04 3.15839663e-04 3.12689868e-04
 3.09857827e-04 3.07204115e-04 3.04809214e-04 3.02609052e-04
 3.00586582e-04 2.99639251e-04 2.99988514e-04]
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$
```

Vertical uncertainty

we calculated:

$$n_{corrected} = n - n_{background}$$

$$\therefore u(n_{corrected}) = \sqrt{u(n)^2 + u(n_{background})^2}$$

where $u(n) = \sqrt{n}$ and ...

$$\therefore u(n_{corrected}) = \sqrt{n + u(n_{background})^2}$$

$u(n_{background})$ is calculated by considering 4×6 minutes experimental runs as a 24 minutes block

The method is as follows:

1. we sum the 4 background radiation run counts
2. then we calculate $\sqrt{n_{background}}$
3. we divide $\sqrt{n_{background}}$ by the number of experimental runs where we counted the background radiation events (for us the number is 4)

```
# uncertainty in the corrected count
corrected_count = count - avg_background_count
u_corrected_count = np.sqrt(count + u_avg_background_count**2)
```

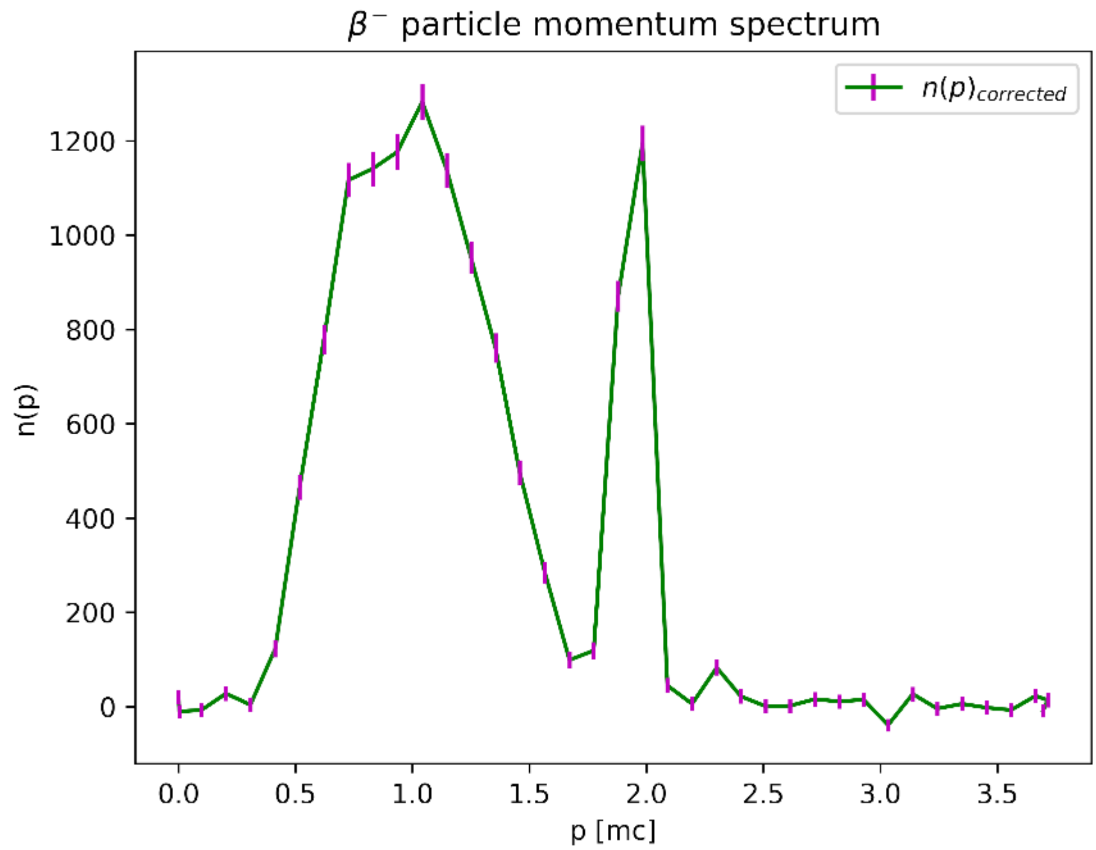
```
background_count = []
# correcting our data by removing avg background count
for row in background_count_data:
    background_count.append(row[5])
avg_background_count = np.mean(background_count)
# print(f"We want to substract thie background count from our data {avg_background_count=}")
# calculating fractional uncertainty in total background count (delta_t = 24 min)
total_background = np.sum(background_count)
u_avg_background_count = np.sqrt(total_background) / 4
```

The variable u_avg_background_count is the expected uncertainty for a 6 minute background radiation events counting run.

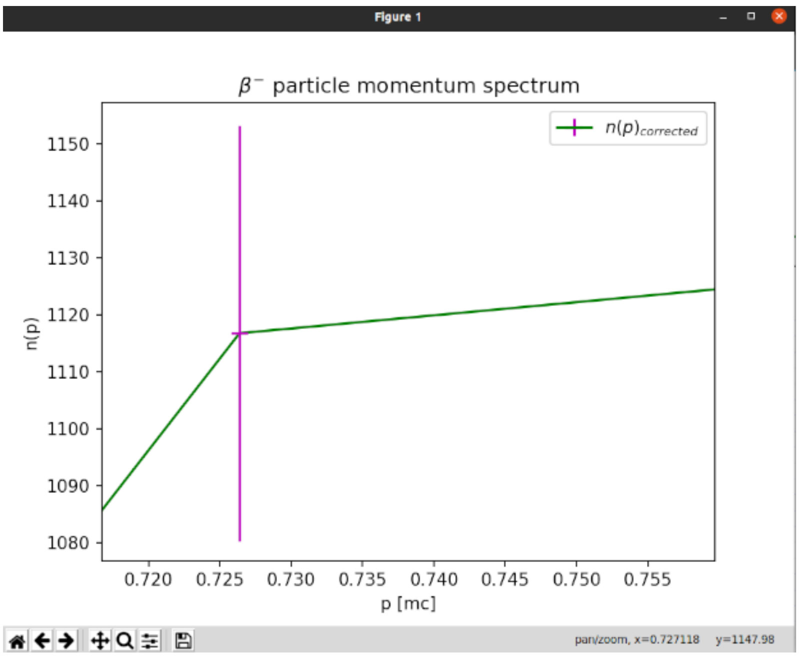
Analysis (Round II)

18/08/2020

— Plot with uncertainties —



Our uncertainty in P is not visible at this scale



This is the zoomed version using matplotlib UI

CODE

```
plt.errorbar(
            p_rel, corrected_count, xerr=u_p_rel, yerr=u_corrected_count,
            marker="None",ecolor="m", label=r"$n(p)_{corrected}$", color="g", barsabove=True
)

plt.title(r"$\beta^{-}$ particle momentum spectrum")
plt.xlabel("p [mc]")
plt.ylabel("n(p)")
plt.legend()
spa.savefig('count_vs_momentum_no_background_error.png')
plt.show()
```

Analysis (Round II)

Kurie Plot

15/08/2020

EQUATIONS

(1) $\omega^2 = p^2c^2 + m_0^2c^4$

(2) $\omega = T + m_0c^2$

(3) $P = p/m_0c$

(4) $W = \omega/m_0c^2$

we derived $W^2 = 1 + P^2$

$$n(p)dp = K_1 F(\overset{\text{constant} = 55}{Z}, \omega)\, p^2 (\omega_0 - \omega)^2 S_n(\omega)\, dp \quad (3)$$

p is momentum of electron

$F(Z,\omega)$ is Fermi function ⇝ accounts for Coulomb attraction: β^- and ${}^{A}_{Z+1}Y$

$Z \to$ charge of ${}^{A}_{Z+1}Y$

$S_n(\omega)$ is shape factor — constant if decay is allowed ∴ e^- carries zero angular momentum; else: forbidden!

$(S_0(\omega) = 1)$

ω_0 is decay energy

∴ $\omega_0 - \omega$ is energy of neutrino

for our analysis we will use a modified Fermi funct.

$$F(Z,\omega) \rightsquigarrow G = \frac{pF(Z=55,\omega)}{\omega}$$

The tabulated data is:

	A	B	C	D	E	F	G
1	Modified Fermi function, G, for Z=55						
2	Note: In this table, "p" is in relativistic units, i.e. unit momentum is m_0 c^2						
3							
4	p	G					
5	0.0	6.591					
6	0.1	6.582					
7	0.2	6.552					
8	0.3	6.506					
9	0.4	6.448					
10	0.5	6.387					
11	0.6	6.329					
12	0.7	6.275					
13	0.8	6.224					
14	0.9	6.177					
15	1.0	6.132					
16	1.2	6.046					
17	1.4	5.964					
18	1.6	5.886					
19	1.8	5.812					
20	2.0	5.742					
21	2.2	5.675					
22	2.4	5.612					
23	2.6	5.553					
24	2.8	5.496					
25	3.0	5.443					

this is a typo relativistic momentum has units mc

We interpolate this data in order to find G for our p's

```
from scipy.interpolate import interp1d
interpolated_fermi_data = interp1d(fermi_data[:,0], fermi_data[:,1],
kind='cubic')
```

We named our function

interpolated_fermi(p)

UNITS

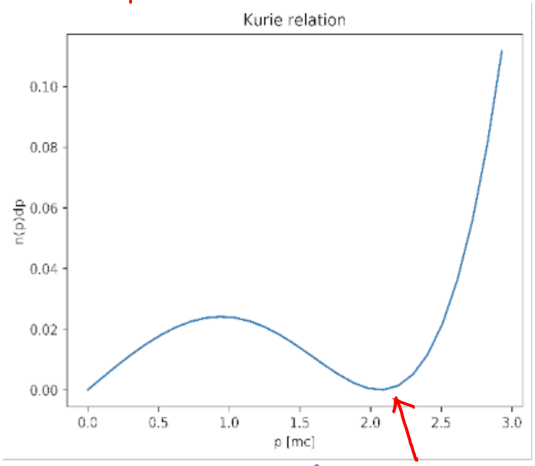
Our momentum data comes from the relation $P = KI$ where $[I]$ = Amps, which is an SI unit. In order to use interpolated_fermi(p) we must convert the units of p from SI into relativistic units. We do this by defining a variable

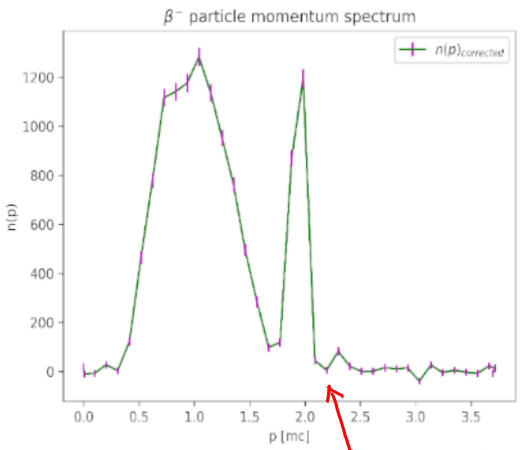
```
rel_energy_unit = mass_e * c**2 # to convert SI into relativistic or viceversa
```

Analysis (Round II)

19/08/2020

— Comparing our plots —



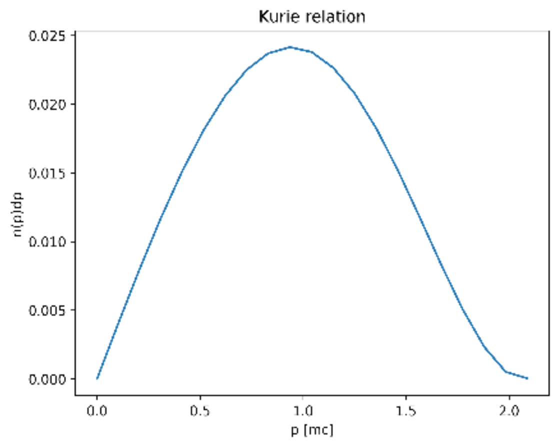


Theoretical n(p)dp minimum when p=p_{max}.

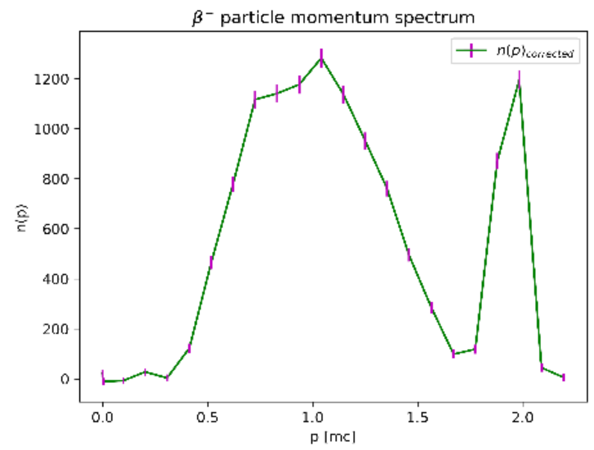
measured minimum for p_{max}

We know that the Kurie relation doesn't take into account the K-peak
in fact the data past the minimum at p_{max} is non physical

∴ we will
exclude these
non-physical points



Vs.



$\bar{p}$[:22] is our
Theoretical sliced range.

```
######################## KURIE PLOT ######################################
dp_rel = p_rel[1]-p_rel[0]

# getting interpolated!
fermi_data = spa.betaray.modified_fermi_function_data
interpolated_fermi = interp1d(fermi_data[:,0], fermi_data[:,1], kind='cubic')

######################## THEORETICAL ################################
# Desintegration energy
# Cs-137 disintegrates by beta minus emission to the ground state of Ba-137 (5,6 %)
w_0 = 1.174 * MeV
w_0_rel = w_0 / rel_energy_unit
p_0_rel = np.sqrt(w_0_rel**2 - 1) / (mass_e * c)
# print(p_0_rel)

# #defining the theoretical count (Kuriefunction)
K_1 = 1 # ?
S_0 = 1
def n(p_rel):
    w_rel = np.sqrt(p_rel**2 + 1) # relativistic energy units
    n = K_1 * S_0* (w_rel * interpolated_fermi(p_rel) / p_rel) * p_rel**2 * (w_0_rel - w_rel)**2
    return n, w_rel

n_p_rel, w_rel = n(p_rel[:22]) #  call and unpack n(p)

# equation (3) in script
N = n_p_rel * dp_rel

plt.figure()
plt.plot(
        p_rel[:22], N, marker="None",
        linestyle="-"
)
plt.title("Kurie relation")
plt.xlabel("p [mc]")
plt.ylabel("n(p)dp")
spa.savefig('Kurie_plot.png')
plt.show()
######################## THEORETICAL ################################
```

```
######################## EXPERIMENTAL ################################

plt.figure()
plt.errorbar(
            p_rel[:23], corrected_count[:23], xerr=u_p_rel[:23], yerr=u_corrected_count[:23],
            marker="None", ecolor="m", label=r"$n(p)_{corrected}$", color="g", barsabove=True
)

plt.title(r"$\beta^{-}$ particle momentum spectrum")
plt.xlabel("p [mc]")
plt.ylabel("n(p)")
plt.legend()
spa.savefig('count_vs_momentum_no_background_error.png')
plt.show()

######################## EXPERIMENTAL ################################
```

$\bar{p}$[:23] is our corresponding
experimental sliced range.

Analysis (Round II)

16/08/2020

Kurie Linearised

$$\boxed{n(\bar{p}) = K_1 F(Z=55, \bar{\omega})\bar{p}^2(\omega_0-\bar{\omega})^2 S_n(\bar{\omega})}$$

written in terms of the interpolated fermi function

$$\left(\frac{n(\bar{p})}{\bar{p}\bar{\omega}G}\right)^{\frac{1}{2}} = K_2 (S_n(\bar{\omega}))^{\frac{1}{2}}(\omega_0-\bar{\omega})$$

in the case of the allowed transition $(S_0(\bar{\omega}) = 1)$

$$\left(\frac{n(\bar{p})}{\bar{p}\omega G}\right)^{\frac{1}{2}} = K_2(\omega_0 - \bar{\omega})$$

19/08/2020

Known uncertainties

- $u(n) = \sqrt{n}$
- $u(p) = p\sqrt{\left(\frac{u(K)}{K}\right)^2 + \left(\frac{u(I)}{I}\right)^2}$

Then

$$\frac{u(\bar{p}^2)}{\bar{p}} = 2\sqrt{\left(\frac{u(K)}{K}\right)^2 + \left(\frac{u(I)}{I}\right)^2}$$

$$\therefore u(\bar{p}^2) = 2\sqrt{\left(\frac{u(K)}{K}\right)^2 + \left(\frac{u(I)}{I}\right)^2}\,\bar{p}$$

since: $\bar{\omega} = \sqrt{\bar{p}+1}$ $\rightsquigarrow$ $u(\bar{\omega}) = \frac{1}{2}\sqrt{(u(\bar{p}^2))^2} = \sqrt{\left(\sqrt{\left(\frac{u(K)}{K}\right)^2 + \left(\frac{u(I)}{I}\right)^2}\,\bar{p}\right)^2}$

$$\therefore u(\bar{\omega}) = u(\bar{p})$$

& since $G = \frac{\bar{p}F}{\omega}$

- $u(G) = \left(\sqrt{\left(\frac{u(\bar{p})}{\bar{p}}\right)^2 + \left(\frac{u(\omega)}{\omega}\right)^2}\right) G$:

```
# uncertainty in interpolated fermi
u_interpolated_fermi = np.sqrt((u_p_rel[:21] / p_rel[:21])**2 + (u_x / x)**2) * interpolated_fermi(p_rel[:21])
```

– LINEAR FIT uncertainty –

$*$ let $y = \left(\frac{n(p)}{p\omega G}\right)^{\frac{1}{2}}$ $*$

Then $\frac{u(y)}{y} = \frac{1}{2}\sqrt{\left(\frac{u(n)}{n}\right)^2 + \left(\frac{u(\bar{p})}{\bar{p}}\right)^2 + \left(\frac{u(\bar{\omega})}{\bar{\omega}}\right)^2 + \left(\frac{u(G)}{G}\right)^2}$ where $\left(\frac{u(\bar{p})}{\bar{p}}\right)^2 + \left(\frac{u(\bar{\omega})}{\bar{\omega}}\right)^2 = 2\left(\frac{u(\bar{p})}{\bar{p}}\right)$

$$\boxed{\therefore \frac{u(y)}{y} = \frac{1}{2}\sqrt{\left(\frac{u(n)}{n}\right)^2 + 2\left(\frac{u(\bar{p})}{\bar{p}}\right)^2 + \left(\frac{u(G)}{G}\right)^2}}$$

we actually have a bit of a problem . . .

Analysis (Round II)

19/08/2020

Kurie Linearised

(CODE model)

$$\left(\frac{n(p)}{pwG}\right)^{\frac{1}{2}} = K_2(\omega_0 - \bar{\omega}) \iff \underbrace{y = K_2(\omega_0 - X)}_{\text{(CODE model)}}$$

– while setting our fit, we obtained this error –

```
ValueError: The model function generated NaN values and the fit aborted! Please check your model function and/or set boundaries on parameters where applicable. In cases like this, using "nan_policy='omit'
" will probably not work.
```

Debbuging Hypothesis: negative numbers causing nan values come from our correction to the count

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 betarays.py
corrected_count[:23]=array([  20.75,  -10.25,   -7.25,   27.75,    3.75,  122.75,  464.75,
        778.75, 1116.75, 1140.75, 1176.75, 1282.75, 1137.75,  951.75,
        760.75,  495.75,  283.75,   98.75,  118.75,  870.75, 1194.75,
         44.75,    5.75])
```

due to Subtraction of the background radiation

• We know that $n(p) \not< 0$

$y =$, $u_y =$

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 betarays.py
betarays.py:160: RuntimeWarning: invalid value encountered in sqrt
  y = np.sqrt(corrected_count[:23] / (p_rel[:23] * x * interpolated_fermi(p_rel[:23])))
array([409.01493712,          nan,          nan,   4.52307902,
         1.3394485 ,   6.52641981,  11.18086471,  12.97994659,
        14.09505136,  13.0479514 ,  12.20325659,  11.80983331,
        10.36390901,   8.88089041,   7.46523865,   5.6870446 ,
         4.06488621,   2.27536798,   2.37526926,   6.13351952,
         6.86791015,   1.27202851,   0.43731927])
array([9.87093819e+03,            nan,            nan, 1.26734468e+00,
       2.63576258e+00, 4.87942754e-01, 3.13576759e-01, 2.62795654e-01,
       2.30431099e-01, 2.10665976e-01, 1.93500952e-01, 1.78207876e-01,
       1.67537398e-01, 1.59355677e-01, 1.53227081e-01, 1.52834293e-01,
       1.59822945e-01, 2.03765591e-01, 1.82454460e-01, 1.16024128e-01,
       1.07909799e-01, 2.28647936e-01, 5.63802484e-01])
```

it's Looking like we found our bug.

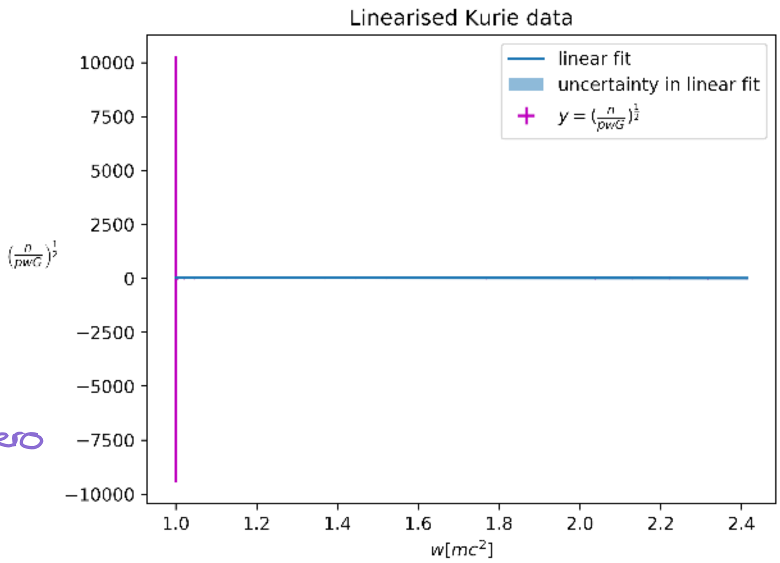
∴ To get rid of these non-physical values of $n(p)$.
We use a Clipped version of our array corrected_counts

```
#this clips negative counts which are non physical
corrected_count = corrected_count.clip(min=0)

y = np.sqrt(corrected_count[:23] / (p_rel[:23] * x * interpolated_fermi(p_rel[:23])))
# regularising y to avoid zero u_y
y_regularised = np.sqrt(corrected_count[:23].clip(min=1) / (p_rel[:23] * x * interpolated_fermi(p_rel[:23])))
u_y = (y_regularised / 2) * np.sqrt((u_corrected_count[:23] / corrected_count[:23].clip(min=1))**2 + (2 * (u_p_rel[:23] / p_rel[:23])**2) + (u_interpolated_fermi / interpolated_fermi(p_rel[:23]))**2)
```

we then use a regularised y in order to calculate our uncertainty

and now our linear fit works!

But it looks like The uncertaintanty is a bit large on the points close to zero

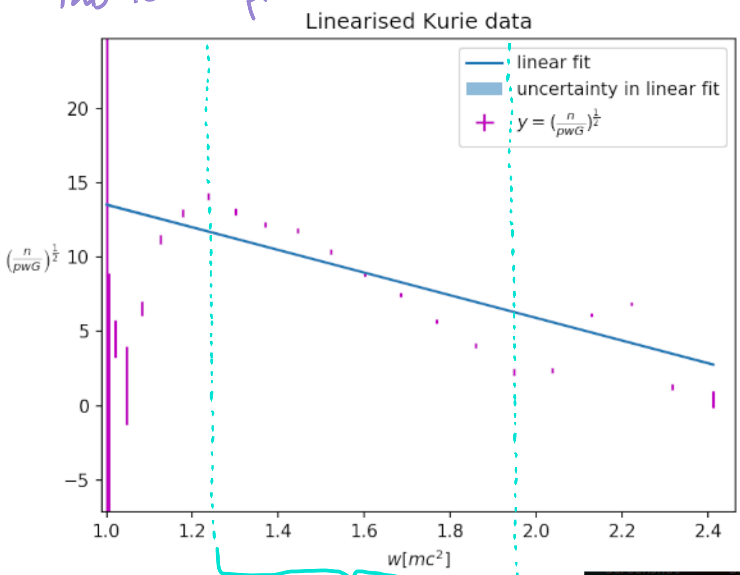


Analysis (Round II)

19/08/2020

It is clear that for an allowed transition, a plot of $\left(\frac{n}{pwG}\right)^{\frac{1}{2}}$ against w yields a straight line intersecting the w-axis at w_0. Forbidden transitions may lead to a non linear Kurie plot, which still yields the correct end point. Providing the non-linearity is not too marked, the end point may still be determined accurately. Determine the end point of your spectrum in this way.

This is our plot zoomed in



The data in this region appears linear

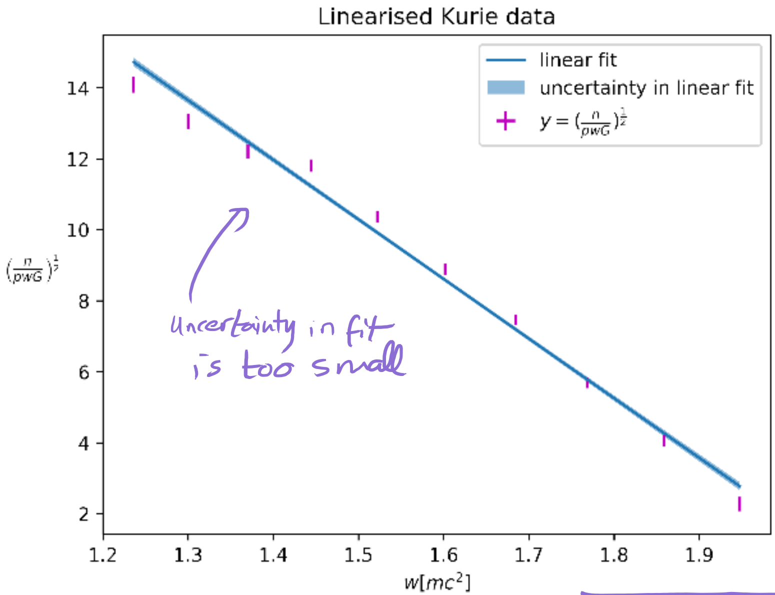
$$\left(\frac{n}{pwG}\right)^{\frac{1}{2}} = K_2(w_0 - \bar{w})$$

$$= K_2 w_0 - K_2 \bar{w}$$

1. from fit get gradient and y-intercept

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 betarays.py
fit_parameters={'slope': -16.787851289977773, 'u_slope': 0.2676432982059722, 'intercept': 35.4717937949
3469, 'u_intercept': 0.4333735111012989}
```

ENHANCE! BY SLICE



$$-K_2 = \text{Slope} = -16.7879$$

$$u(\text{slope}) = 0.2676$$

$$\therefore K_2 = 16.7879 \pm 0.2676$$

$$K_2 w_0 = \text{Intercept} = 35.4718$$

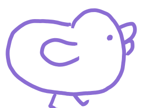
$$u(\text{intercept}) = 0.4334$$

$$\therefore w_0 = \frac{\text{Intercept}}{K_2}$$

$$\therefore u(w_0) = \sqrt{\left(\frac{u(K_2)}{K_2}\right)^2 + \left(\frac{u(\text{Intercept})}{\text{Intercept}}\right)^2}\; w_0$$

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 betarays.py
fit_parameters={'slope': -16.787851289977773, 'u_slope': 0.2676432982059722, 'in
tercept': 35.47179379493469, 'u_intercept': 0.4333735111012989}
K_2=16.787851289977773
w_0=2.1129442465404193
u_w_0=0.0424398995686856
w_0_rel=2.297460725203179
```

← we calculated $w_0 = \frac{1.174 \text{ MeV}}{mc^2}$



Sceptical duck

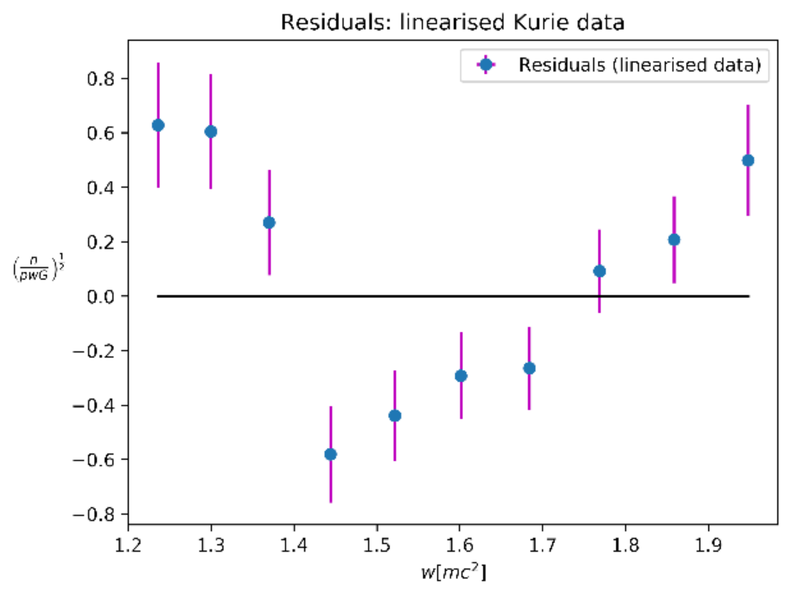
$$\therefore w_0 = 2.11 \pm 0.04$$

Analysis (Round II)

19/08/2020

Residuals

As expected, it doesn't look too good!



Comparison to theory

20/08/2020

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 -i  betarays.py
fit_parameters={'slope': -16.787851289977773, 'u_slope': 0.2676432982059722, 'intercept': 35.47179379493469, 'u_intercept': 0.4333735111012989}
K_2=16.787851289977773
w_0=2.1129442465404193
u_w_0=0.0424398995686856
w_0_rel=2.297460725203179
>>> diff = w_0_rel - w_0
>>> print(diff)
0.18451647866275955
>>> diff / u_w_0
4.347712424816989
>>>
```

$\frac{w_{0\,rel} - w_0}{\sigma} =$

Our result is 4.35σ away from the expected result.

it appears that we have underestimated our uncertainty

SciPy.org — To optimise our fit.

SciPy.org / Docs / SciPy v1.5.2 Reference Guide / Optimization and root finding (scipy.optimize)

scipy.optimize.curve_fit

scipy.optimize.curve_fit(f, xdata, ydata, p0=None, sigma=None, absolute_sigma=False, check_finite=True, bounds=(-inf, inf), method=None, jac=None, **kwargs) [source]

Use non-linear least squares to fit a function, f, to data.

Assumes ydata = f(xdata, *params) + eps.

Returns:

popt : *array*
Optimal values for the parameters so that the sum of the squared residuals of f(xdata, *popt) - ydata is minimized.

pcov : *2-D array*
The estimated covariance of popt. The diagonals provide the variance of the parameter estimate. To compute one standard deviation errors on the parameters use perr = np.sqrt(np.diag(pcov)).

Scipy.optimize.curve_fit

requires a model $f = mx + c$

The parameters
$m \rightarrow K_2 = 16.7879 \pm 0.2676$
$x \rightarrow \bar{w} \pm u(\bar{p})$
$c \rightarrow K_2 w_0 = \text{Intercept} = 35.4718 \pm 0.4334$

we need to propagate uncertainty to obtain $u(f)$
as we know

$$u(f(x,m,c))^2 = \left(\frac{\partial f}{\partial x} u(x)\right)^2 + \left(\frac{\partial f}{\partial m} u(m)\right)^2 + \left(\frac{\partial f}{\partial c} u(c)\right)^2$$

$$\therefore u(f(x,m,c)) = \sqrt{(m\,u(x))^2 + (x\,u(m))^2 + (u(c))^2}$$

CODE

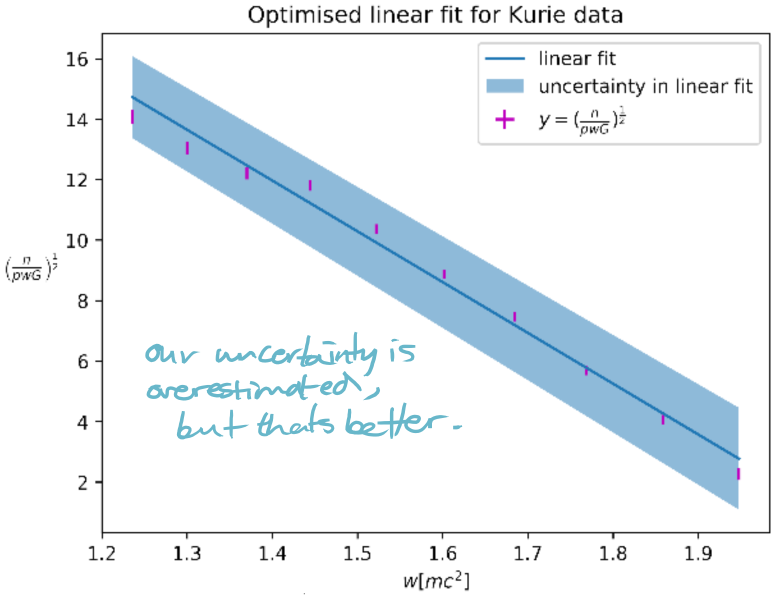
```python
# linear model for optimize.curve_fit()
def f(x, m, c):
    return m * x + c
# uncertainty in linear model f
u_f = np.sqrt((K_2 * u_x)**2 + (x * u_K_2)**2 + (u_intercept)**2)

# optimising our fit, unpack into popt, pcov
popt, pcov = scipy.optimize.curve_fit(f, x, y, sigma=u_y, absolute_sigma=False)
# To compute one standard deviation errors on the parameters use
perr = np.sqrt(np.diag(pcov))

opt_K_2, opt_intercept = popt
u_opt_K_2, u_opt_intercept = perr
```

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 betarays.py
linear fit gradient: K_2 = 16.787851289977773
linear fit intercept: intercept = 35.47179379493469

THIS IS OUR EXPECTED RESULT theory_w_0_rel = 2.297460725203179
THIS IS OUR RESULT w_0 = 2.1129442465404193 ± 0.0424398995686856

optimised gradient -16.787851225186017 ± 0.6656759006640471
optimised intercept 35.47179369119481 ± 1.0778760504058646
```

Analysis (Round II)

20/08/2020

Optimised fit plot.

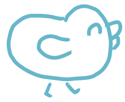


our uncertainty is overestimated, but thats better.

```
# using our results to find opt_w_0
# recall that:
# opt_K_2, opt_intercept = popt
# u_opt_K_2, u_opt_intercept = perr
opt_w_0 = opt_intercept / opt_K_2
u_opt_w_0 = np.sqrt((u_opt_K_2 / - opt_K_2)**2 + (u_opt_intercept / opt_intercept)**2) * opt_w_0

print(f"EXPECTED RESULT {theory_w_0_rel = }")
print(f"post-optimisation result  {opt_w_0 = } ± {u_opt_w_0}\n")
```

```
ana@ana-XPS-13-9343:~/Documents/uni/PHS3000$ python3 -i betarays.py
linear fit gradient: K_2 = 16.787851289977773
linear fit intercept: intercept = 35.47179379493469

EXPECTED RESULT theory_w_0_rel = 2.297460725203179
pre-optimisation result  w_0 = 2.1129442465404193 ± 0.0424398995686856

optimised gradient -16.787851225186017 ± 0.6656759006640471
optimised intercept 35.47179369119481 ± 1.0778760504058646

EXPECTED RESULT theory_w_0_rel = 2.297460725203179
post-optimisation result  opt_w_0 = 2.112944248515746 ± 0.10555548613871778

>>> diff = theory_w_0_rel - opt_w_0
>>> diff / u_opt_w_0
1.7480519813526991
>>>
```

← 1.74σ away from the value

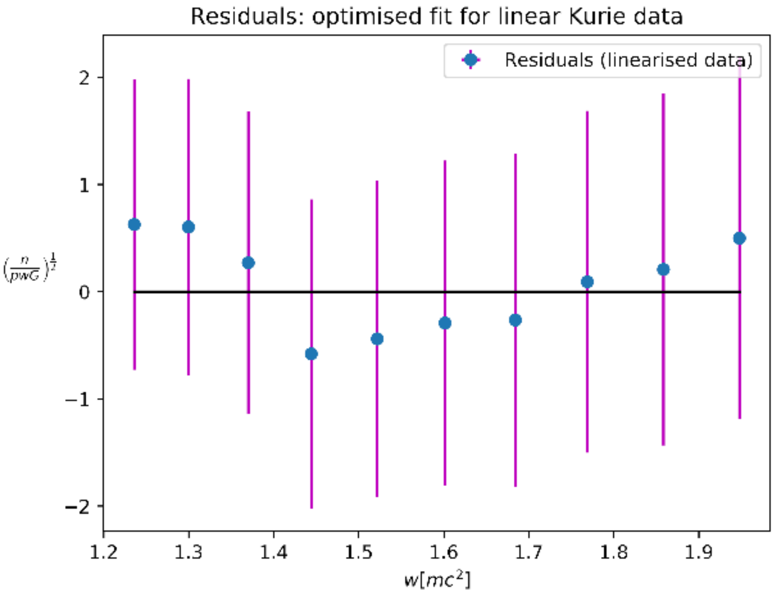


$\therefore W_0 = 2.11 \pm 0.11 \ mc^2$

we still don't have agreement but our result makes more sense

— non-relativistic result —

$$\boxed{W_0 = 1.08 \pm 0.05 \text{ MeV}}$$



Siegbahn $S_n(\bar{W})$

$$S_n(\bar{W}) = \bar{W}^2 - 1 + (W_0 - \bar{W})^2 \ [16]$$

is the best fit if $\Delta j = \pm 2$ (First forbidden)

The Shape factor

Investigate the results of using the shape factor given in Siegbahn, page 497.

✓ From your determined end point energy, calculate the corresponding beta particle kinetic energy, and compare to the accepted value of the (maximum) beta particle energy for ^{137}Cs.

$$\left(\frac{n}{pwG}\right)^{\frac{1}{2}} = K_2 (S_n(\bar{w}))^{\frac{1}{2}} (w_0 - \bar{w})$$

$$= K_2 \sqrt{\bar{W}^2 - 1 + (W_0 - \bar{W})^2} \ (w_0 - \bar{w})$$

$$= K_2 w_0 \sqrt{\bar{w}^2 - 1(w_0 - \bar{w})^2} - K_2 \bar{w} \sqrt{\bar{w}^2 - 1(w_0 - \bar{w})^2}$$

This relation is non-linear anymore we would need to write a new fit in order to use it to analyse our data

This might be more comprehensive than our linearised Kurie relation

Discussion

20/08/2020

The experimental process was bumpy and complicated.
We had very little clue of what was to be expected when operating the apparatus, but also our early understanding of the theory was incomplete.
This is very visible when we read the method we followed. during our first attempt at running the apparatus on (06/08/2020)
We expected that learning about the control interface was going to be a more efficient process.

Our aim during 1 RUN was to understand the experimental control interface, this aim was reasonable but our naïve approach made the process very slow. We arbitrarily chose our starting current to be 1.000A (0.9932±0.0005 A)
This choice had problematic repecussions, since the momentum spectrum was incomplete. Our initial choice in the increments to the current was not too ideal given the range of the current available in the apparatus.
We chose to run the experiment in constant time. This choice remained throughout every run we planned. Our choice to set every count to be (600.000±0.015s) proved to be non-ideal given that we ran out of time during our first booking with the experimental apparatus.
As we already implied, the controls interface was highly unintuitive during 1 RUN we noticed that our set current was recorded as zero valued every time we set a half integer value for the lens coil current.
We decided to continue with 1 RUN without modification to the method.
We aimed to solve the observed problems during 2 RUN.

During 2 RUN our first goal was to calibrate the magnetometer probe, After this we started investigating how to operate the bias coil current and its effect in the recording of the magnetic field.
We set the value of the bias coil current to 0.190A (0.1890±0.0005 A) and noticed that only the y-component of the field approached zero.
We where unable to understand a key part of our experimental controls at this stage
The ENABLE button activated the current, but if pressed again it would deactivate the current too.
Because of this "small" misunderstanding we observed the same problem with our record of the current. Everytime we made an increment we would press the enable button, tragically deactivating the current.
We understood this during 2 RUN.
This discovery lead us to understand how to attempt to cancel the magnetic field of earth.

3 RUN our choice for the initial lens coil current to be 1.000A (0.9934±0.0005 A) remained, but this time we where able to get a record of every increment made to our lens coil current. We decided to set our increments to be 0.250 A in order to obtain a more comprehensive spectrum.
After disabling the lens coil current we set the value of the bias coil current to 0.830A which allowed us to minimize earth's magnetic field.

— unsolved problems on (06/08/2020) —

- During every run, we chose an inadequate range for our current (1A to 3.5A)
- We were unable to come with an idea to figure out the count of the background radiation.

- We plotted our result and learned about the inadequacy of our range but unfortunately we ran out of time during this session.

This experience yielded NO analysis.

Discussion

20/08/2020

After all of these issues, we had another opportunity to expand in our theoretical understanding of the experiment.

The relations between current, momentum and rigidity we obtained from Siegbahn ($R = B\rho$) and Quade, Halliday ($p = kI$) allowed us to determine the proportionality constant between momentum, current and the magnetic field originated in the apparatus.

Since the Rigidity is a measure of a particle's momentum
higher rigidity ⟺ higher momentum.

We decided that it was important for us to reduce the running time for the count of each current.
We made it a priority that we should only count for as long as necessary to maintain a count uncertainty of less than 5%.

Experimental Procedure (Round II)

We were more organised this time around, in addition during our educational hiatus we came up with a way to determine the background count.

We decided that we would use current increment values of 0.1A to maximise our chance at finding an adequately shaped momentum spectrum.

To minimise earth's magnetic field we set the bias current to 0.720 A (0.716 ± 0.0005 A). Similarly to before, we set our experiment to run in constant time, and before starting (1) RUN we proceeded to close the shutter for the Cs-137 source in order to record the background radiation for 6 minutes. (we did this 4 times).

(1) RUN was better planned on 18/08, this allowed us to have a more automatic process outlined. We did not encounter issues for any of the counts, we started counting from the range: [0, 3.6 A]

where 0A = 0.0000 ± 0.0005 A and 3.6000A = 3.5245 ± 0.0005 A.

We were able to calibrate our momentum spectrum with the expected K-peak and we were able to correct our count for an average of the 4 recorded counts of the background radiation.

The background correction led us to encounter an error with the numerical evaluation of negative counts inside a square root which had to be ammended by clipping our un-physical negative counts.
We proceeded to calculate a non zero uncertainty in the clipped values by defining a "regularised" value for the Linearized count as per the model in equation 4.

Our linear fit was succesful after these corrections but the momentum data had to be sliced to a smaller apparently linear range as was demonstrated by the plots in our analysis.

Since all uncertainties were propagated correctly to the best of our knowledge. We decided to try our hand at optimisation.

Discussion

20/08/2020

The linear fit produced a result with an unreasonalely small uncertainty. This lead us to try an optimisation function based on a Least squares fit (from scipy library)

we used a linear model, our energy data, and Linear count data as inputs to this optimisation function.

The function can be configured to use only the relative magnitudes of uncertainties and scaling them by a constant factor that is set by imposing that $\chi^2_{red} = 1$.

This by definition improves the fit.

This means that our uncertainties are scaled to match the sample variance residuals after our fit occurs.

This certainly improved the scale of our uncertainty in the result for the total released energy. putting our value 1.7σ away from the true value for Cs-137

our result after optimisation is $w_0 = 1.08 \pm 0.05$ MeV.

Reading The docs

near verbatim from reference [17]

Conclusion

20/08/2020

Using a magnetic spectrometer we measured the energy spectrum of electrons emitted by β^- radiation from a Cs-137 source. Our investigation focused in determining the relationships between current, momentum and energy.

We found that the total energy released by the observed nuclear transition was $W_0 = 1.08 \pm 0.05$ MeV.
Our result disagrees with the expected theoretical result.

```python
# PHS3000 - LOGBOOK1
# Betarays - radioactive decay Cs - 137
# Ana Fabela, 15/08/2020
import monashspa.PHS3000 as spa
from scipy.interpolate import interp1d
import numpy as np
import pandas as pd
import pytz
import matplotlib
import matplotlib.pyplot as plt
from pprint import pprint
import scipy.optimize

plt.rcParams['figure.dpi'] = 150

# Globals
c = 299792458 # [m/s]
mass_e = 9.10938356e-31 # [kg]
eV = 1.602176634e-19 # [J]
MeV = 1e6 * eV
keV = 1e3 * eV
rel_energy_unit = mass_e * c**2 # to convert SI into relativistic or viceversa

data = spa.betaray.read_data(r'beta-ray_data.csv')

# valid data slicing from csv file
j = 0
for row in data:
    # print(f'{j=}')
    if row[0] == pd.Timestamp('2020-08-18 10:26:00+10:00', tz=pytz.FixedOffset(360)):
        valid_data = data[j:]
        continue
    j+=1

background_count_data = []
count = []
lens_current = []
u_lens_current = []
for row in valid_data:
    if row[3] == 'Closed':
        # print(row[3])
        background_count_data.append(row)
        continue
    count.append(row[5])
    lens_current.append(row[6])
    u_lens_current.append(row[7])

background_count = []
# correcting our data by removing avg background count
for row in background_count_data:
    background_count.append(row[5])
avg_background_count = np.mean(background_count)
# print(f"We want to substract thie background count from our data {avg_background_count=}")
# calculating fractional uncertainty in total background count (delta_t = 24 min)
total_background = np.sum(background_count)
u_avg_background_count = np.sqrt(total_background) / 4

# uncertainty in the corrected count
corrected_count = count - avg_background_count
u_corrected_count = np.sqrt(count + u_avg_background_count**2)


# Finding constant of proportionality in p = kI
# calibration peak (K) index of k peak is i=20
T_K = 624.21 * keV / rel_energy_unit
k = np.sqrt((T_K + 1)**2 - 1) / lens_current[20]
# print(f"{k=}")
u_k = k * (0.0005 / lens_current[20])
# print(f"absolute uncertainty: {u_k = }")
# print(f"fractional uncertainty: {(u_k / k) = }\n")

# The momentum spectrum
lens_current = np.array(lens_current)
p_rel = k * lens_current
u_p_rel = p_rel * np.sqrt((u_k / k)**2 + (0.0005 / lens_current)**2)
# print(f"absolute uncertainty u(p_rel):\n {u_p_rel}")
# print(f"fractional uncertainty u(p_rel) / p_rel:\n {(u_p_rel / p_rel)}")

# plot
plt.figure()
plt.errorbar(
            p_rel, corrected_count, xerr=u_p_rel, yerr=u_corrected_count,
            marker="None", ecolor="m", label=r"$n(p)_{corrected}$", color="g", barsabove=True
)

plt.title(r"$\beta^{-}$ particle momentum spectrum")
plt.xlabel("p [mc]")
plt.ylabel("n(p)")
plt.legend()
spa.savefig('count_vs_momentum_no_background_error.png')
plt.show()

###################### KURIE/Fermi PLOT ####################################

dp_rel = p_rel[1]-p_rel[0]

# getting interpolated!
fermi_data = spa.betaray.modified_fermi_function_data
interpolated_fermi = interp1d(fermi_data[:,0], fermi_data[:,1], kind='cubic')

####################### THEORETICAL ###############################
# Desintegration energy
# Cs-137 disintegrates by beta minus emission to the ground state of Ba-137 (5,6 %)
theory_w_0 = 1.174 * MeV
theory_w_0_rel = theory_w_0 / rel_energy_unit
p_0_rel = np.sqrt(theory_w_0_rel**2 - 1) / (mass_e * c)
# print(p_0_rel)

# # defining the theoretical count (Kuriefunction)
K_1 = 1 # ?
Sn = 1
def n(p_rel):
    w_rel = np.sqrt(p_rel**2 + 1) # relativistic energy units
    n = K_1 * Sn * (w_rel * interpolated_fermi(p_rel) / p_rel) * p_rel**2 * (theory_w_0_rel - w_rel)**2
    return n, w_rel


n_p_rel, w_rel = n(p_rel[:22]) #  call and unpack n(p)

# equation (3) in script
N = n_p_rel * dp_rel

# plot
plt.figure()
plt.plot(
        p_rel[:22], N, marker="None",
        linestyle="-"
)
plt.title("Kurie relation")
plt.xlabel("p [mc]")
plt.ylabel("n(p)dp")
spa.savefig('Kurie_plot.png')
plt.show()

####################### THEORETICAL ###############################
###################### EXPERIMENTAL ###############################

# plot
plt.figure()
plt.errorbar(
            p_rel[:23], corrected_count[:23], xerr=u_p_rel[:23], yerr=u_corrected_count[:23],
            marker="None", ecolor="m", label=r"$n(p)_{corrected}$", color="g", barsabove=True
)

plt.title(r"$\beta^{-}$ particle momentum spectrum")
plt.xlabel("p [mc]")
plt.ylabel("n(p)")
plt.legend()
spa.savefig('count_vs_momentum_no_background_error.png')
```

```python
plt.show()

###################### EXPERIMENTAL ##############################
###################### KURIE/Fermi PLOT ####################################
########################## Linear fit ###########################
# initial slice [:23]
# second slice [8:18]
n_p_rel, w_rel = n(p_rel[8:18])

# our sliced data linearised
x = w_rel
u_x = u_p_rel[8:18]

# uncertainty in interpolated fermi
u_interpolated_fermi = np.sqrt((u_p_rel[8:18] / p_rel[8:18])**2 + (u_x / x)**2) * interpolated_fermi(p_rel[8:18])

# this clips negative counts which are non physical
corrected_count = corrected_count.clip(min=0)

# LINEARISED KURIE
y = np.sqrt(corrected_count[8:18] / (p_rel[8:18] * x * interpolated_fermi(p_rel[8:18])))
# regularising y to avoid zero u_y
y_regularised = np.sqrt(corrected_count[8:18].clip(min=1) / (p_rel[8:18] * x * interpolated_fermi(p_rel[8:18])))
u_y = (y_regularised / 2) * np.sqrt((u_corrected_count[8:18] / corrected_count[8:18].clip(min=1))**2 + (2 * (u_p_rel[8:18] / p_rel[8:18])**2) + (u_interpolated_fermi / interpolated_fermi(p_rel[8:1

fit_results = spa.linear_fit(x, y, u_y=u_y)
# making our linear fit with one sigma uncertainty
y_fit = fit_results.best_fit
u_y_fit = fit_results.eval_uncertainty(sigma=1)

# calculating values from fit results
fit_parameters = spa.get_fit_parameters(fit_results)
# print(f"{fit_parameters=}")

# using our results to find w_0
K_2 = - fit_parameters["slope"]
u_K_2 = fit_parameters["u_slope"]
intercept = fit_parameters["intercept"]
u_intercept = fit_parameters["u_intercept"]
w_0 = intercept / K_2
u_w_0 = np.sqrt((u_K_2 / K_2)**2 + (u_intercept / intercept)**2) * w_0

print(f"linear fit gradient: {K_2 = }")
print(f"linear fit intercept: {intercept = }\n")

print(f"EXPECTED RESULT {theory_w_0_rel = }")
# pre-optimisation result
print(f"pre-optimisation result  {w_0 = } ± {u_w_0}\n")

# plot
plt.figure()
plt.errorbar(
            x, y, xerr=u_p_rel[8:18], yerr=u_y,
            marker="None", linestyle="None", ecolor="m",
            label=r"$y = (\frac{n}{p w G})^{\frac{1}{2}}$", color="g", barsabove=True
)
plt.plot(
        x, y_fit, marker="None",
        linestyle="-",
        label="linear fit"
)
plt.fill_between(
                x, y_fit - u_y_fit,
                y_fit + u_y_fit,
                alpha=0.5,
                label="uncertainty in linear fit"
)
plt.title("Linearised Kurie data")
plt.xlabel(r"$w [mc^{2}]$")
plt.ylabel(r"$\left ( \frac{n}{p w G} \right )^{\frac{1}{2}}$", rotation=0, labelpad=18)
plt.legend()
spa.savefig('Kurie_linear_data_plot_.png')
plt.show()

############################ Linear fit ############################
#########################Linear fit residuals######################

linear_residuals = y_fit - y # linear residuals (linear best fit - linearised data)

# plot
plt.figure()
plt.errorbar(
            x, linear_residuals, xerr=u_p_rel[8:18], yerr=u_y,
            marker="o", ecolor="m", linestyle="None",
            label="Residuals (linearised data)"
)
plt.plot([x[0], x[-1]], [0,0], color="k")
plt.title("Residuals: linearised Kurie data")
plt.xlabel(r"$w [mc^{2}]$")
plt.ylabel(r"$\left ( \frac{n}{p w G} \right )^{\frac{1}{2}}$", rotation=0, labelpad=18)
plt.legend()
spa.savefig('linear_residuals_Kurie_linear_data.png')
plt.show()

# #########################Linear fit residuals######################

# linear model for optimize.curve_fit()
def f(x, m, c):
    return m * x + c

# optimising our fit, unpack into popt, pcov
popt, pcov = scipy.optimize.curve_fit(f, x, y, sigma=u_y, absolute_sigma=False)
# To compute one standard deviation errors on the parameters use
perr = np.sqrt(np.diag(pcov))

opt_K_2, opt_intercept = popt
u_opt_K_2, u_opt_intercept = perr

print(f"optimised gradient {opt_K_2} ± {u_opt_K_2}")
print(f"optimised intercept {opt_intercept} ± {u_opt_intercept}\n")

optimised_fit = f(x, opt_K_2, opt_intercept)
# uncertainty in linear model f given optimal fit
u_f = np.sqrt((opt_K_2 * u_x)**2 + (x * u_opt_K_2)**2 + (u_opt_intercept)**2)

# using our results to find opt_w_0
opt_w_0 = opt_intercept / - opt_K_2
u_opt_w_0 = np.sqrt((u_opt_K_2 / opt_K_2)**2 + (u_opt_intercept / opt_intercept)**2) * opt_w_0

print(f"EXPECTED RESULT {theory_w_0_rel = }")
print(f"post-optimisation result  {opt_w_0 = } ± {u_opt_w_0}\n")
print(f"non-relativistic w_0 = {opt_w_0 * rel_energy_unit / MeV} ± {u_opt_w_0 * rel_energy_unit / MeV}\n")

# OPTIMISED FIT PLOT
plt.figure()
plt.errorbar(
            x, y, xerr=u_p_rel[8:18], yerr=u_y,
            marker="None", linestyle="None", ecolor="m",
            label=r"$y = (\frac{n}{p w G})^{\frac{1}{2}}$", color="g", barsabove=True
)
plt.plot(
        x, optimised_fit, marker="None",
        linestyle="-",
        label="linear fit"
)
plt.fill_between(
                x, optimised_fit - u_f,
                optimised_fit + u_f,
                alpha=0.5,
                label="uncertainty in linear fit"
)
plt.title("Optimised linear fit for Kurie data")
plt.xlabel(r"$w [mc^{2}]$")
plt.ylabel(r"$\left ( \frac{n}{p w G} \right )^{\frac{1}{2}}$", rotation=0, labelpad=18)
plt.legend()
spa.savefig('OPTIMISED_Kurie_linear_data_plot_.png')
plt.show()

#########################optimised fit residuals######################
```

```python

optimised_residuals = optimised_fit - y
# plot
plt.figure()
plt.errorbar(
            x, optimised_residuals, xerr=u_p_rel[8:18], yerr=u_f,
            marker="o", ecolor="m", linestyle="None",
            label="Residuals (linearised data)"
)
plt.plot([x[0], x[-1]], [0,0], color="k")
plt.title("Residuals: optimised fit for linear Kurie data")
plt.xlabel(r"$w [mc^{2}]$")
plt.ylabel(r"$\left ( \frac{n}{p w G} \right )^{\frac{1}{2}}$", rotation=0, labelpad=18)
plt.legend()
spa.savefig('OPTIMISED_linear_residuals_Kurie_linear_data.png')
plt.show()

# # ##########################optimised fit residuals#######################
```

Logbook 1: References

[0] Monash SPA (2020) 4.4 Measurement of β-ray spectra. Retrieved from Moodle (student site)

[1] Wikipedia (2020). Beta decay. Retrieved from https://en.wikipedia.org/wiki/Beta_decay

[2] Wikipedia (2020).Nuclide. Retrieved from https://en.wikipedia.org/wiki/Nuclide

[3] Wikipedia (2020). Isobar. Retrieved from https://en.wikipedia.org/wiki/Isobar_(nuclide)

[4] Nave (2001). Hyperphysics: Energy and momentum spectra for Beta decay. Retrieved from http://hyperphysics.phy-astr.gsu.edu/hbase/Nuclear/beta2.html

[5] Wikipedia (2019). Internal conversion. Retrieved from https://en.wikipedia.org/wiki/Internal_conversion

[6] Witcher (1941) An Electron Lens Type of Beta-Ray Spectrometer.pdf Retrieved from (4.4 Measurement of β-ray spectra: References)

[7] Bell, *et al.* (2015). The Effect of the Earth's and Stray Magnetic Fields on Mobile Mass Spectrometer Systems. https://doi.org/10.1007/s13361-014-1027-4

[8] Krane, Section 2.7(also 2.8);12.8 (also 12.6 pp.386-387). http://phet.colorado.edu/en/simulation/beta-decay

[9] Siegbahn, (1968) Beta Ray Spectrometer Theory and Design pp79-201.pdf Retrieved from (4.4 Measurement of β-ray spectra: References)

[10] Quade, EA Halliday, D (1947) Quade, EA Halliday, D 1947.pdf Retrieved from (4.4 Measurement of β-ray spectra: References)

[11] Wikipedia (2019). Rigidity (electromagnetism). Retrieved from https://en.wikipedia.org/wiki/Rigidity_(electromagnetism)

[12] R.G.HelmerandV.P.Chechev (2006)Cs-137_tables. Retrieved from http://www.nucleide.org/DDEP_WG/Nuclides/Cs-137_tables.pdf

[13] Wikipedia (2020). Chi-square distribution. Retrieved from https://en.wikipedia.org/wiki/Chi-square_distribution

[14] Chris Billington (2020) victoria.py. Retrieved from https://github.com/chrisjbillington/chrisjbillington.github.io/blob/master/victoria.py

[15] Wikipedia (2020). Propagation of uncertainty Retrieved from https://en.wikipedia.org/wiki/Propagation_of_uncertainty

[16] Siegbahn, (1968) Beta Ray Spectrometer vol1_pp496-498 K Siegbahn.pdf Retrieved from (4.4 Measurement of β-ray spectra: References)

[17] The SciPy community (2020) scipy.optimize_curve. Retrieved from https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html