# beta ray end

November 29, 2021

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
import warnings

warnings.filterwarnings("ignore")
```

## 1 Datas

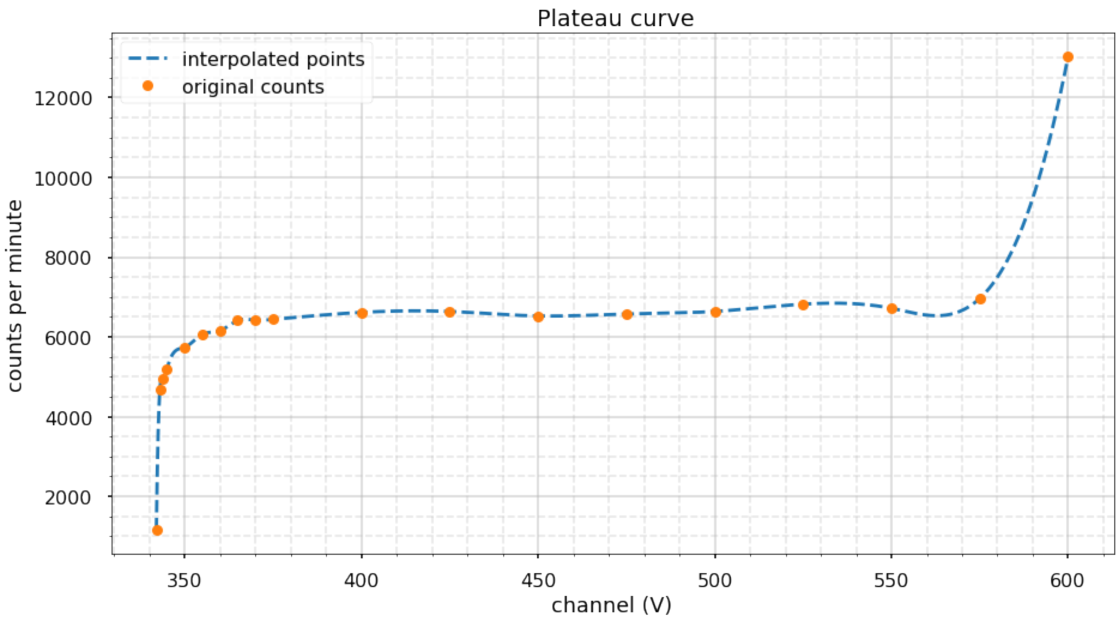
[2]:
```python
# plateau datas
data_plateau = pd.read_excel("endpoint_datas.xlsx", sheet_name="plateau")
pl_voltage = data_plateau["p_voltage"]
pl_counts = data_plateau["p_counts"]

# thickness data
data_thickness_tl = pd.read_excel("endpoint_datas.xlsx", sheet_name="tl")
data_thickness_sryt = pd.read_excel("endpoint_datas.xlsx", sheet_name="sryt")
# tellurium datas
tl_thickness_original = data_thickness_tl["t_thickness_tl"]
tl_counts_original = data_thickness_tl["t_counts_tl"]
# strontium-ytterium datas
sryt_thickness_original = data_thickness_sryt["s_thickness_sryt"]
sryt_counts_original = data_thickness_sryt["s_counts_sryt"]

# distances data
data_distance = pd.read_excel("endpoint_datas.xlsx", sheet_name="distance")
# tellurium
distance_tl = data_distance["d_distance"]
counts_tl = data_distance["d_counts_tl"]
# strontium-ytterium
distance_sryt = data_distance["d_distance"]
counts_sryt = data_distance["d_counts_sryt"]
```

## 2 Functions

[3]:
```
# function for interpolation
def interpolate(x, y):
    f = interp1d(x, y, kind="cubic", fill_value="extrapolate")
    a = np.arange(x[0], x[len(x) - 1], 0.001)
    b = f(a)
    return a, b


# funciton for polynomial fitting
def polfit(a, b, c):
    z = np.polyfit(a, b, c)
    f = np.poly1d(z)

    x = np.arange(a[0], a[len(a) - 1], 0.001)
    y = f(x)
    return x, y
```

## 3 Plateau

[4]:
```
voltage_interpolated_pl, counts_interpolated_pl = interpolate(pl_voltage,␣
↪pl_counts)

plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))
plt.title(f" Plateau curve")
plt.xlabel("channel (V)")
plt.ylabel("counts per minute")
plt.plot(voltage_interpolated_pl, counts_interpolated_pl, "--",␣
↪label="interpolated points")
plt.plot(pl_voltage, pl_counts, "o", markersize=9, label="original counts")
plt.legend(loc="upper left")
plt.grid(alpha=0.5, which="major")
plt.minorticks_on()
plt.grid(alpha=0.3, which="minor", ls="--")
plt.show()
```




I choosed the operating voltage at 400 V

## 4 Thickness Curve

Varying the absorber in the GM counter tube.

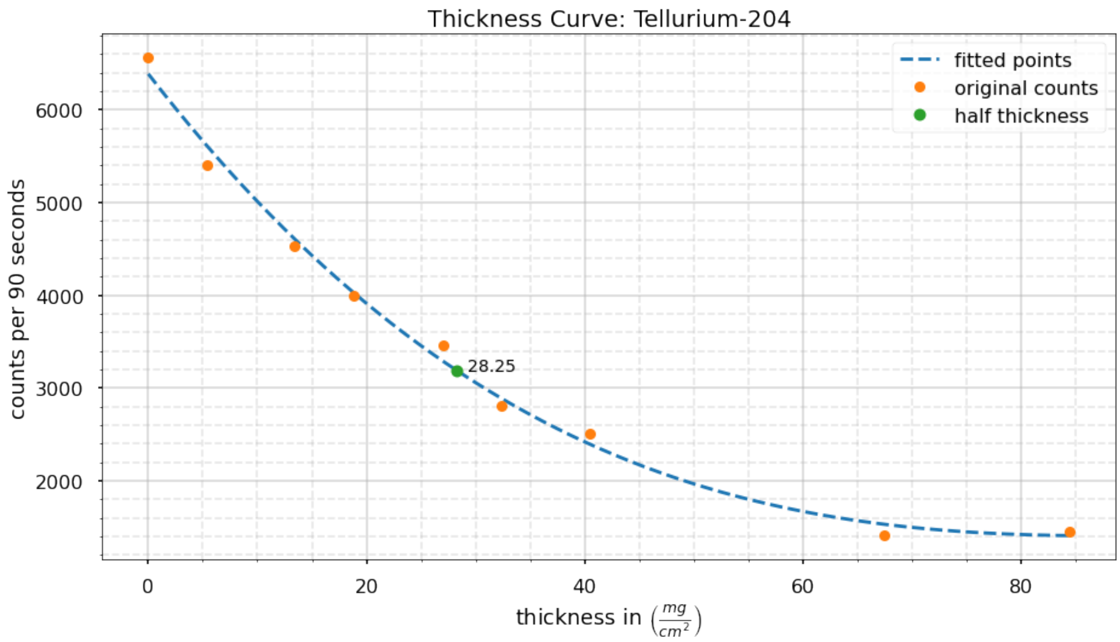
[5]:
```
thickness_fitted_tl, counts_fitted_tl = polfit(tl_thickness_original,␣
↪tl_counts_original, 3)
thickness_fitted_sryt, counts_fitted_sryt = polfit(sryt_thickness_original,␣
↪sryt_counts_original, 3)

element_name = ["Tellurium-204", "Strontium-90 Yttrium-90"]
thickness_fitted = [thickness_fitted_tl, thickness_fitted_sryt]
counts_fitted = [counts_fitted_tl, counts_fitted_sryt]
thickness_original = [tl_thickness_original, sryt_thickness_original]
counts_original = [tl_counts_original, sryt_counts_original]

# finding the half width
thickness_half = []
c_half = []
for i in range(len(element_name)):
    count_half = counts_fitted[i][0] / 2
    c_half.append(count_half)
    th = interp1d(counts_fitted[i], thickness_fitted[i], kind="cubic")
    thickness_half.append(th(count_half))
    print(
```

```
        f"{element_name[i]}: \n\t max count = {counts_fitted[i][0]:.0f}, half␣
↪count = {count_half:.0f} \n\t half thickness = {thickness_half[i]:.2f} cm"
    )
```

```
Tellurium-204:
         max count = 6386, half count = 3193
         half thickness = 28.25 cm
Strontium-90 Yttrium-90:
         max count = 1523, half count = 761
         half thickness = 97.05 cm
```

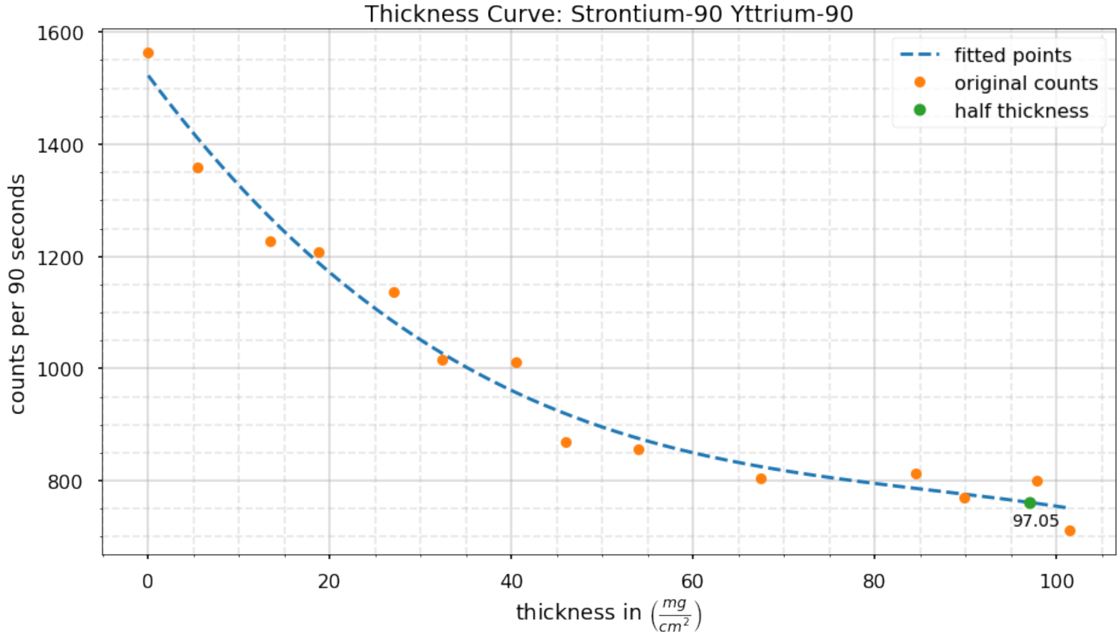
### 4.1 Tellurium

[6]:
```
# plotting the curves
plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))

plt.title(f"Thickness Curve: {element_name[0]}")
plt.xlabel(r"thickness in $\left(\frac{mg}{cm^2}\right)$")
plt.ylabel("counts per 90 seconds")

plt.plot(thickness_fitted[0], counts_fitted[0], "--", label="fitted points")
plt.annotate(f"{thickness_half[0]:.2f}", xy=(thickness_half[0] + 1, c_half[0]),␣
↪fontsize=14)
plt.plot(thickness_original[0], counts_original[0], "o", markersize=9,␣
↪label="original counts")
plt.plot(thickness_half[0], c_half[0], "o", markersize=10, label="half␣
↪thickness")

plt.legend(loc="upper right")
plt.grid(alpha=0.5, which="major")
plt.minorticks_on()
plt.grid(alpha=0.3, which="minor", ls="--")

plt.show()
```




## 4.2 Strontium-Yetrrium

[7]:
```
plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))


plt.title(f"Thickness Curve: {element_name[1]}")
plt.xlabel(r"thickness in $\left(\frac{mg}{cm^2}\right)$")
plt.ylabel("counts per 90 seconds")

plt.plot(thickness_fitted[1], counts_fitted[1], "--", label="fitted points")
plt.annotate(f"{thickness_half[1]:.2f}", xy=(thickness_half[1] - 2, c_half[1] -␣
↪40), fontsize=14)
plt.plot(thickness_original[1], counts_original[1], "o", markersize=9,␣
↪label="original counts")
plt.plot(thickness_half[1], c_half[1], "o", markersize=10, label="half␣
↪thickness")

plt.legend(loc="upper right")
plt.grid(alpha=0.5, which="major")
plt.minorticks_on()
plt.grid(alpha=0.3, which="minor", ls="--")

plt.show()
```




### 4.3 Calculation using the half thickness

[8]:
```
#  units used is mg/cm^2
t1 = thickness_half[0]
t2 = thickness_half[1]
r1 = 291.083

# using the relation t1/t2 = r1/r2
r2 = r1 * (t2 / t1)
print(f"the range of Sr-Yt  = {r2:.3f} mg/cm^2")

# n = 1.265 - 0.0954* np.log(e2)
# r2 = 412 * e2**n

coeff = [-0.0954, 1.265, -(np.log(r2 / 412))]
solution = np.roots(coeff)
for i in range(len(solution)):
    if solution[i] <= 1:
        energy = np.exp(1) ** solution[i]
        print(f"the energy of Sr-Yt = {energy:.3f} MeV")
```

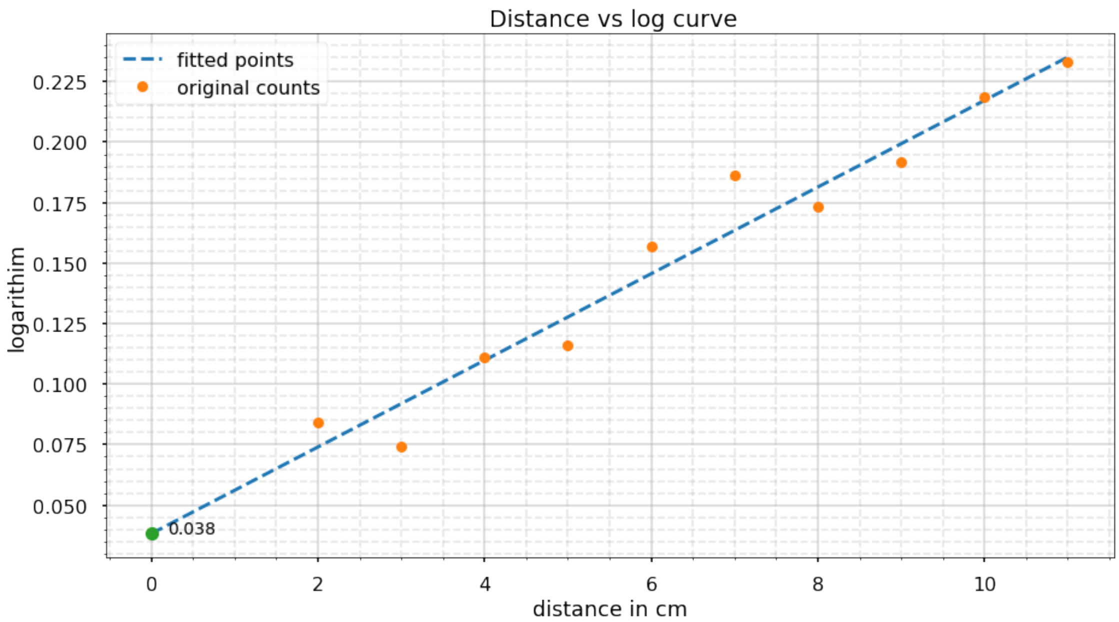
```
the range of Sr-Yt  = 1000.111 mg/cm^2
the energy of Sr-Yt = 2.102 MeV
```

## 5 Distance

varying the distance of the source from the GM counter. Done outside the tube in a moveable scale and holder

[11]:
```
# function for interpolation
def inter0(x, y):
    f = interp1d(x, y, kind="linear", fill_value="extrapolate")
    a = np.arange(0, x[len(x) - 1], 0.001)
    b = f(a)
    return a, b


n_01 = 22968
n_02 = 4061

# logf = np.log10((counts_tl / counts_sryt) * 0.1768)
logf = np.log10((counts_sryt / counts_tl) * (n_01/ n_02))
dist = data_distance["d_distance"]

dist_fit, logf_fit = polfit(dist, logf, 1)

d, l = inter0(dist_fit, logf_fit)

# c = np.interp(0, dist_fit, logf_fit)
c = np.interp(0, d, l)
print(f"the intercept = {c:.3f}")


plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))
plt.title(f" Distance vs log curve")
plt.xlabel("distance in cm")
plt.ylabel("logarithim")
# plt.plot(dist_fit, logf_fit, "--", label="fitted points")
plt.plot(d, l, "--", label="fitted points")
plt.plot(dist, logf, "o", markersize=9, label="original counts")
plt.plot(0, c, "o")
plt.annotate(f"{c:.3f}", xy=(0.2, c), fontsize=14)

plt.grid(alpha=0.5, which="major")
plt.minorticks_on()
plt.grid(alpha=0.3, which="minor", ls="--")

plt.legend(loc="upper left")
plt.show()
```

```
the intercept = 0.038
```




## 5.1 final calculation

[10]:
```
mu_1 = 17 * (0.764 ** -1.43)
mu_2 = 17 * (2.102 ** -1.43)

del_mu = mu_1 - mu_2

w = c / del_mu
print(f"the thickness = {w:.5f} mg/cm^2")
```

```
the thickness = 0.00200 mg/cm^2
```

[ ]:
```

```