Geiger

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
```

read data

```python
data = pd.read_csv("geiger.csv")["counts"].to_numpy()
data[:10]
```

```
array([213, 217, 233, 226, 202, 232, 207, 206, 214, 221])
```

get mean counts and bg

```python
# Since we are woring with rates over time
# We scale most errors by (time_err/measured_time)
time_err = 0.01

mean_counts = data.mean()
mean_err = np.sqrt(data.mean())
mean_time = 10
mean_count_rate = mean_counts/mean_time
mean_count_rate_err = mean_count_rate*np.sqrt(
  (mean_err/mean_counts)**2 + (time_err/mean_time)
)
bg_counts = 100
# we set it up so that this was the fractional error
bg_counts_err = 0.1
bg_time = 320.87
bg_count_rate = bg_counts/bg_time
bg_count_rate_err = bg_count_rate*np.sqrt(
  (bg_counts_err/bg_counts)**2 + (time_err/bg_time)
)
print(f"mean_counts: {mean_counts}+-{mean_err:.4f}")
print(f"bg_count_rate: {bg_count_rate:.4e}+-{bg_count_rate_err:.4e} s^-1")
print(f"mean_count_rate: {mean_count_rate:.4f}+-{mean_count_rate_err:.4f} s^-1")
```

```
mean_counts: 216.74+-14.7221
bg_count_rate: 3.1165e-01+-1.7675e-03 s^-1
mean_count_rate: 21.6740+-1.6239 s^-1
```

```python
# Define symbolic variables
counts = sp.symbols('c1 c2 c12')
errors = sp.symbols('sigma1 sigma2 sigma12')
c1, c2, c12 = counts
sigma1, sigma2, sigma12 = errors

# Define equations
t_dead = (c1 + c2 - c12) / (2 * c1 * c2)
delta_t_dead = sp.sqrt(sum(d**2 * sp.diff(t_dead,c)**2 for c,d in zip(counts, errors)))

# Define numeric values
time_rel_error = time_err/100
vals = {
  c1: 112.15,
  c2: 105.92,
  c12: 182.43,
  # Errors are values scaled by relative error in time measurement
  sigma1: time_rel_error*112.15,
  sigma2: time_rel_error*105.92,
  sigma12: time_rel_error*182.43,
}

# Display info
print("Dead Time")
display(t_dead)
print("Dead Time Error")
display(delta_t_dead)

print(f"t_dead = {t_dead.subs(vals):.4e} s")
print(f"delta_t_dead = {delta_t_dead.subs(vals):.4e} s")
```

```
Dead Time
```

$$\frac{c_1 - c_{12} + c_2}{2c_1c_2}$$

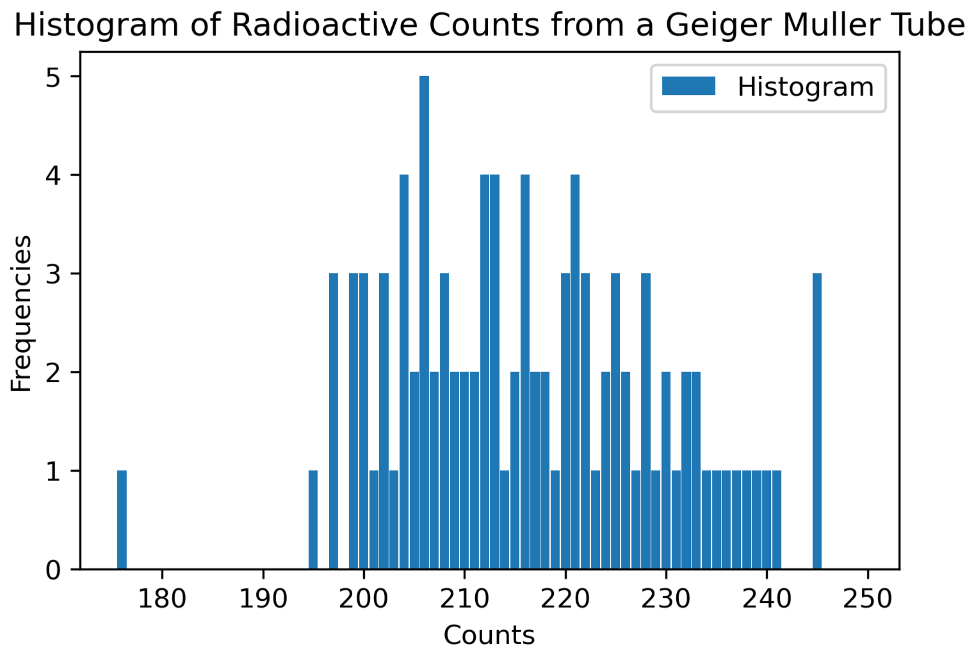
```
Dead Time Error
```

$$\sqrt{\sigma_1^2 \left(\frac{1}{2c_1c_2} - \frac{c_1 - c_{12} + c_2}{2c_1^2c_2} \right)^2 + \sigma_2^2 \left(\frac{1}{2c_1c_2} - \frac{c_1 - c_{12} + c_2}{2c_1c_2^2} \right)^2 + \frac{\sigma_{12}^2}{4c_1^2c_2^2}}$$

```
t_dead = 1.5001e-3 s
delta_t_dead = 8.8366e-7 s
```

Plot poisson distribution

```python
bin_bounds = np.arange(min(data),max(data)+1,1)-0.5
plt.figure()
plt.hist(data, bin_bounds, label="Histogram", rwidth=0.90)
plt.title("Histogram of Radioactive Counts from a Geiger Muller Tube")
plt.xlabel("Counts")
plt.ylabel("Frequencies")
plt.legend()
plt.show()
```



Get the source counting rate

```
source_count_rate = mean_count_rate - bg_count_rate
source_count_rate_err = np.sqrt(mean_count_rate_err**2 + bg_count_rate_err**2)
print(f"source_count_rate = {source_count_rate:.4f}+-{source_count_rate_err:.4f} s^-1")
```

```
source_count_rate = 21.3623+-1.6239 s^-1
```