spark

December 5, 2021

[44]:
```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.interpolate import interp1d
import warnings
from scipy import optimize


warnings.filterwarnings("ignore")
```

1 Functions

[45]:
```python
# # segment fitting of data
# def segments_fit(X, Y, count):
#     xmin = X.min()
#     xmax = X.max()

#     seg = np.full(count - 1, (xmax - xmin) / count)

#     px_init = np.r_[np.r_[xmin, seg].cumsum(), xmax]
#     py_init = np.array([Y[np.abs(X - x) < (xmax - xmin) * 0.01].mean() for x␣
↪in px_init])

#     def func(p):
#         seg = p[:count - 1]
#         py = p[count - 1:]
#         px = np.r_[np.r_[xmin, seg].cumsum(), xmax]
#         return px, py

#     def err(p):
#         px, py = func(p)
#         Y2 = np.interp(X, px, py)
#         return np.mean((Y - Y2)**2)

#     r = optimize.minimize(err, x0=np.r_[seg, py_init], method='Nelder-Mead')
#     return func(r.x)
```

```python
# def cur_fit(x,y):
#     func = lambda t, a, c, d: a*np.log(t + c) + d
#     popt, pcov = optimize.curve_fit(func, x, y, np.array([0.5,0.5,0.5]))
#     xx = np.arange(x[0], x[len(x) - 1], 0.001)
#     yy = func(xx, *popt)
#     return xx, yy


# function for interpolation
def interpolate(x, y):
    f = interp1d(x, y, kind="quadratic", 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
```

2 Importing Datas

[46]:
```python
# need to get the values of voltage and current
data_current = pd.read_excel("spark_data.xlsx", sheet_name="vol_current")
voltage_0 = data_current["voltage"]
current_0 = data_current["current"]

# voltage vs. counts
data_counts_1 = pd.read_excel("spark_data.xlsx", sheet_name="vol_count1")
voltage_1 = data_counts_1["voltage_d1"]
counts_1 = data_counts_1["counts_d1"]

data_counts_2 = pd.read_excel("spark_data.xlsx", sheet_name="vol_count2")
voltage_2 = data_counts_2["voltage_d2"]
counts_2 = data_counts_2["counts_d2"]

data_counts_3 = pd.read_excel("spark_data.xlsx", sheet_name="vol_count3")
voltage_3 = data_counts_3["voltage_d3"]
counts_3 = data_counts_3["counts_d3"]

# distance vs counts
```

```
data_distance = pd.read_excel("spark_data.xlsx", sheet_name="dist_count")
distance = data_distance["distance"]
counts = data_distance["counts"]
```

3 Current characteristics

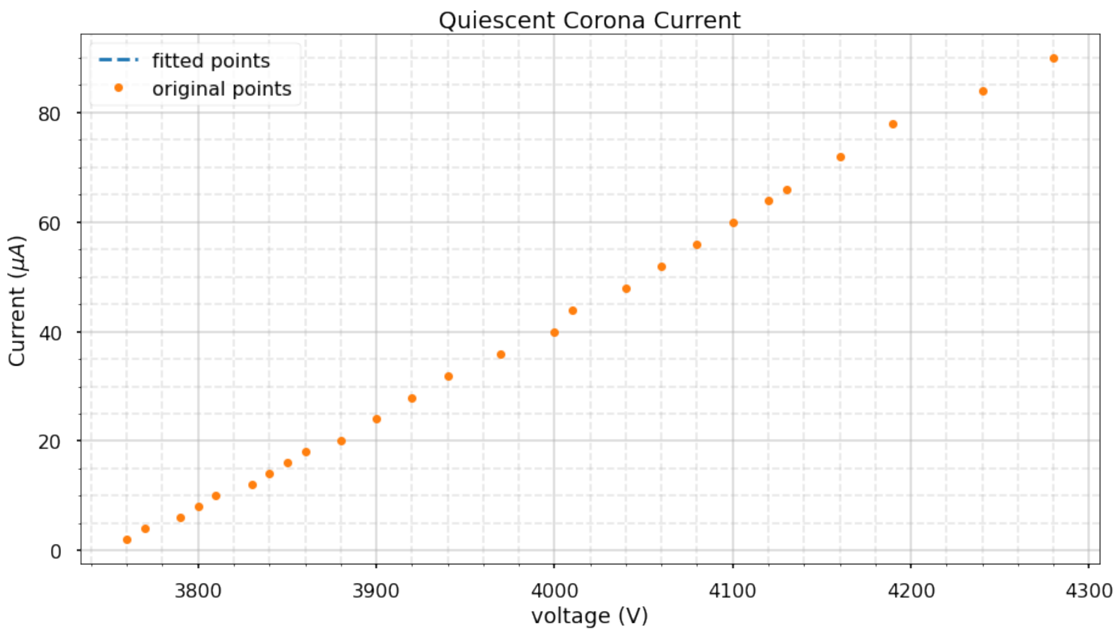
[47]:
```
def curr(x,y):
        for i in range(len(x)):
                if (y[i] <= 4000):
                        x_int, y_int = interpolate(x,y)
                        return x_int, y_int
                else:
                        x_int, y_int = polfit(x,y,1)
                        return x_int, y_int

current_0_interpolated,voltage_0_interpolated = interpolate(current_0,␣
 ↪voltage_0)
```

[54]:
```
plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))
plt.title(f"Quiescent Corona Current")
plt.xlabel("voltage (V)")
plt.ylabel(r"Current $\left(\mu A\right)$")

ad, bd = [],[]
# plt.plot(voltage_0_interpolated, current_0_interpolated, "--",␣
 ↪label="inteprolated points")
plt.plot(ad, bd, "--", label="fitted points")
plt.plot(voltage_0, current_0, "o", markersize=7, label=f"original points")

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()
```



4 Voltage Characteristics

4.1 interpolation

[49]:
```python
# original datas
voltage_orginal = [voltage_1, voltage_2, voltage_3]
counts_original = [counts_1, counts_2, counts_3]

# interpolation
voltage_interpolated_1, counts_interpolated_1 = interpolate(voltage_1, counts_1)
voltage_interpolated_2, counts_interpolated_2 = interpolate(voltage_2, counts_2)
voltage_interpolated_3, counts_interpolated_3 = interpolate(voltage_3, counts_3)

# order = 50
# voltage_interpolated_1, counts_interpolated_1 = polfit(voltage_1, counts_1,␣
 ↪order)
# voltage_interpolated_2, counts_interpolated_2 = polfit(voltage_2, counts_2,␣
 ↪order)
# voltage_interpolated_3, counts_interpolated_3 = polfit(voltage_3, counts_3,␣
 ↪order)

# interpolated datas
voltage_interpolated = [voltage_interpolated_1, voltage_interpolated_2,␣
 ↪voltage_interpolated_3]
counts_interpolated = [counts_interpolated_1, counts_interpolated_2,␣
 ↪counts_interpolated_3]
```

4.2 plot

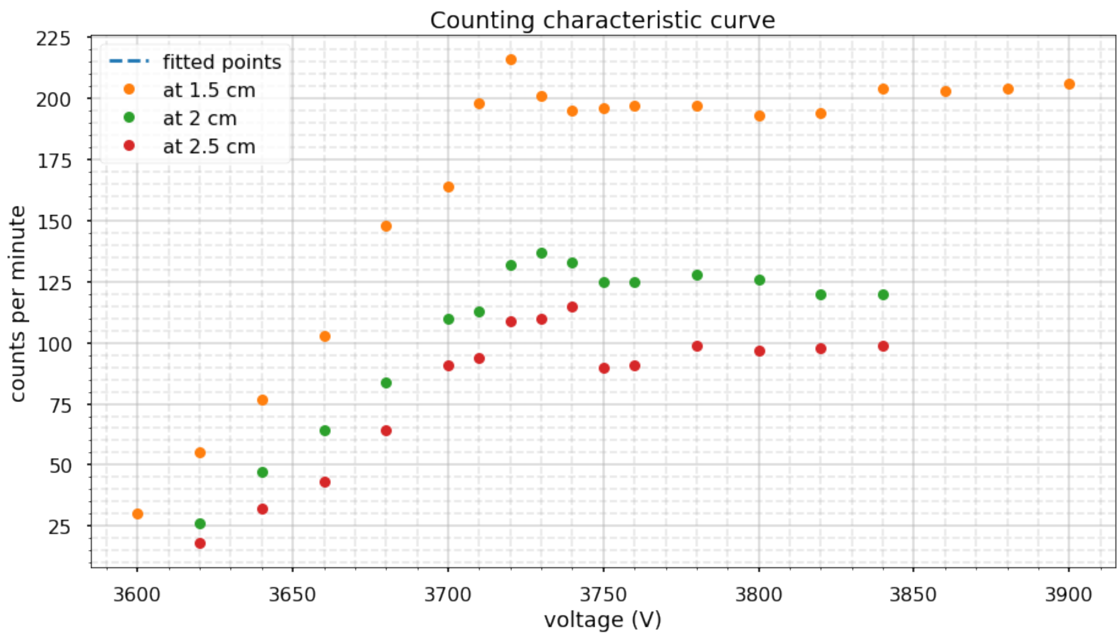
[56]:
```python
dilen = [1.5, 2, 2.5]

plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))
plt.title(f"Counting characteristic curve")
plt.xlabel("voltage (V)")
plt.ylabel("counts per minute")
px,py = [],[]
plt.plot(px, py,"--", label ="fitted points")
for i in range(len(voltage_interpolated)):
    # px, py = segments_fit(voltage_interpolated[i], counts_interpolated[i], 3)
    # px,py = polfit(voltage_interpolated[i], counts_interpolated[i], 5)

    # plt.plot(voltage_interpolated[i], counts_interpolated[i], "--")
    # pxint, pyint = polfit(px,py,4)

    # plt.plot(pxint, pyint, "--")
    plt.plot(
        voltage_orginal[i], counts_original[i], "o", markersize=9, label=f"at␣
 ↪{dilen[i]} cm"
    )
```

```
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()
```

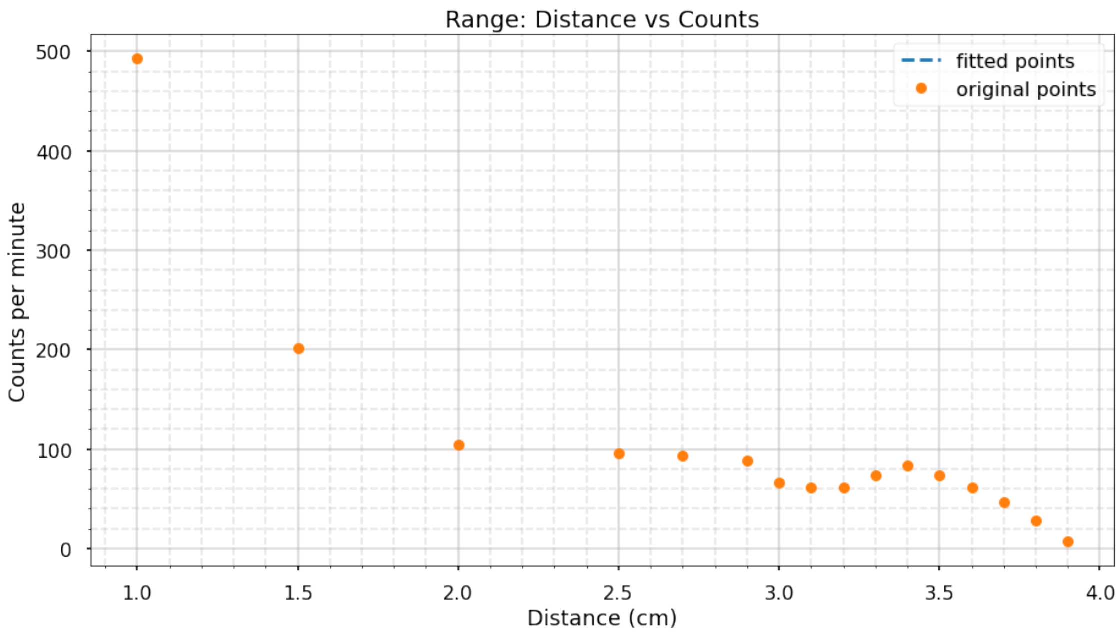


5 Distance vs counts

[51]:
```
plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))
plt.title(f"Range: Distance vs Counts")
plt.xlabel("Distance (cm)")
plt.ylabel(f"Counts per minute")

ad, bd = [],[]
# ad, bd = polfit(distance, counts, 5)
plt.plot(ad, bd, "--", label="fitted points")
plt.plot(distance,counts, "o", markersize=9, label=f"original points")

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()
```


Range: Distance vs Counts
fitted points
original points
Counts per minute
Distance (cm)
0
100
200
300
400
500
1.0
1.5
2.0
2.5
3.0
3.5
4.0