Mo32_Photoeffekt_v2

June 2, 2024

1 Fakultät für Physik

1.1 Physikalisches Praktikum P2 für Studierende der Physik

Versuch P2-63, 64, 65 (Stand: April 2024)

Raum F1-08

2 Photoeffekt

Name: Vrkic Vorname: Tin E-Mail: uyvpq@student.kit.edu

Name: Nock Vorname: Mika E-Mail: uttzi@student.kit.edu

Gruppennummer: Mo-32

Betreuer: Denis Benâtre

Versuch durchgeführt am: 27.05.24

Beanstandungen zu Protokoll Version _____:

Testiert am: _____ Testat: _____

[ ]:
```
# importieren aller benötigten Module
import matplotlib.pyplot as plt
import numpy as np
import kafe2
import pathlib
import pandas as pd
from uncertainties import ufloat

# erstellen einer Funktion für kafe2 Fits
def fit_funktion(xy_data, model_function, xy_error, xy_label, title,␣
 ↪constraint=[]):
    xy_data = kafe2.XYContainer(xy_data[0], xy_data[1])
    xy_data.label = title
    fit = kafe2.XYFit(xy_data = xy_data, model_function = model_function)
    fit.add_error(axis = 'x', err_val = xy_error[0])
    fit.add_error(axis = 'y', err_val = xy_error[1])
    for i in range(len(constraint)):
        fit.add_parameter_constraint(name = constraint[i][0], value =␣
 ↪constraint[i][1], uncertainty = constraint[i][2])
    fit.do_fit()
    plot = kafe2.Plot(fit)
    plot.x_label, plot.y_label = xy_label[0], xy_label[1]

    return fit.parameter_values, fit.parameter_errors, plot

e = 1.602176634 * 10**(-19) # elemental charge in Coulomd
c = 2.99792458 * 10**8 # speed of light in meters/second
```

2.1 Aufgabe 1: Grundlagen

Hinweise zu Aufgabe 1 finden in der Datei Hinweise-Versuchsdurchfuehrung.md.

- Machen Sie sich mit dem zu untersuchenden Effekt und der Art, wie Sie ihn beobachten und messen werden vertraut.
- Bearbeiten Sie hierzu die folgenden Aufgaben.

2.1.1 Aufgabe 1.1: Qualitative Beobachtung des äußeren photoelektrischen Effekts

Beobachten Sie den äußeren Photoeffekt mit Hilfe des bereitstehenden statischen Elektrometers (E), der Zn-Platte (Zn) und der Hg-Dampflampe (Hg) qualitativ. Gehen Sie dabei wie folgt vor:

- Laden Sie Zn negativ auf und beobachten Sie E ohne Zn mit Hg zu bestrahlen.
- Laden Sie Zn negativ auf und beobachten Sie E, wenn Sie Zn mit Hg bestrahlen.
- Laden Sie Zn negativ auf und beobachten Sie E, wenn Sie Zn mit Hg bestrahlen und zusätzlich eine positiv geladene Elektrode in die Nähe von Zn bringen.
- Laden Sie Zn positiv auf und beobachten Sie E, wenn Sie Zn mit Hg bestrahlen.

Beschreiben und erklären Sie Ihre Beobachtungen.

After charging up the electrometer, the following things happen in the cases mentioned above

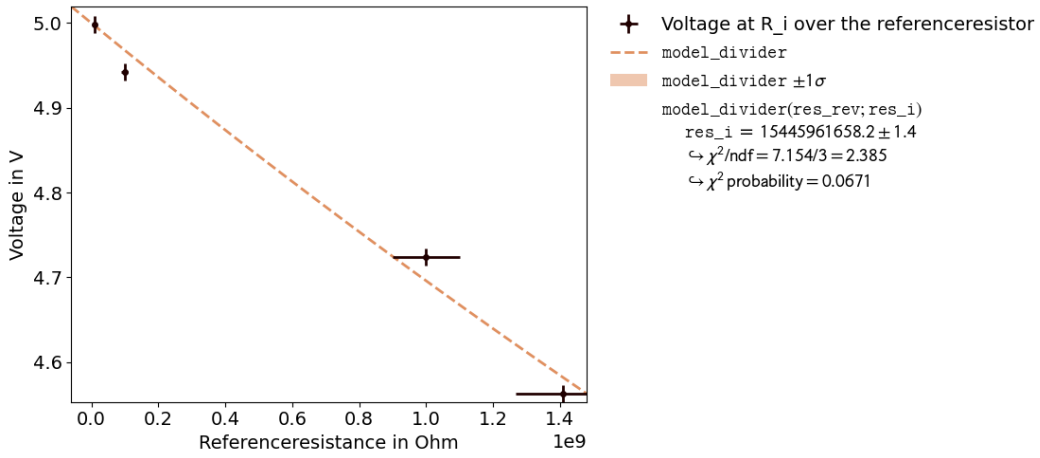
- With a negatively charged plate, the electrons leak a little into the environment (for example in coronal discharges) and the electrometer discharges very slowly over time. The half-life was so high we couldn't measure it appropriately.
- With a irradiation, the electrometer discharged faster, with a half-life of about $2\,min$. That is the case because electrons will be knocked out of the plate by the incoming photons and more of them can escape into the environment.
- Approaching the plate with a positively charged electrode makes the electrometer discharge much faster again because the electrons that get knocked out by the photons are attracted by the electrode and don't form a charged cloud. The half-life in this case was about $20\,s$.
- When the electrometer is positively charged, it isn't affected by the incoming photons. That's because there are no more electrons that can be knocked out by the photons and the charge is held.

2.1.2 Aufgabe 1.2: Charakterisierung des für die folgenden Aufgaben zu verwendenden Elektrometers

- Nehmen Sie einen Nullabgleich des für **Aufgabe 2** zu verwendenden Elektrometers vor.
- Bestimmen Sie den Innenwiderstand R_i des Elektrometers mit Hilfe der vorhandenen Vorwiderstände von $R_V = 0.01,\ 0.1,\ 1$ und $10\,\mathrm{G}\,$.

[ ]:
```
def model_divider(res_rev,res_i):
    return U_0 * res_i / (res_rev+res_i)

res_rev = np.array([10,100,1000,10000]) * 10**6 #in Ohm
```

```
U_e = np.array([4.998,4.942,4.724,4.563])
U_0 = 5

xy_data = kafe2.XYContainer(res_rev,U_e)
xy_data.label = 'Voltage at R_i over the referenceresistor'
fit = kafe2.XYFit(xy_data = xy_data, model_function = model_divider)
fit.add_error('x', err_val = 0.1, relative = True)
fit.add_error('y', err_val = 0.01)
fit.do_fit()

plot = kafe2.Plot(fit)
plot.x_label = "Referenceresistance in Ohm"
plot.y_label = "Voltage in V"
plot.plot()
plt.show()
```



The elctrometer was adjusted to zero for an amplification of 10^2. The main problem were fluctuations that were hard to control and the fact that an adjustment for values over 10^2 didn't hold when the amplifier was turned back down to lower amplifications. Instead it showed values of around $4\,V$.

The inner resistance of the mesuring device was determined with a voltage divider. Some known resistors were plugged in series to the inner resistance and the voltagedrop across the inner resistor was mesured. With a fit, the final result can then be obtained. In this case, the model used was $U_e = U_0 \cdot \frac{R_i}{R_{rev}+R_i}$ which follows from Kirchhoff's laws.

The Resistance is calculated to $R_i = 1.54 \cdot 10^{10}\Omega$, which is about three orders of magnitude different from the actual value $R_i \geq 10^{13}$ written on the electrometer. It can't be said why this error occurs. It is enough, though, to say that the inner resistance is very high, which is the important aspect.

2.2 Aufgabe 2: Bestimmung von h

Hinweise zu Aufgabe 1 finden in der Datei Hinweise-Versuchsdurchfuehrung.md.

- Bestimmen Sie h aus dem äußeren photoelektrischen Effekt mit Hilfe von zwei verschiedenen Schaltungen.
- Bearbeiten Sie hierzu die folgenden Aufgaben.

2.2.1 Aufgabe 2.1: Spannung U_{Ph} der Photozelle bei variierender Lichtfrequenz

- Tragen Sie die sich von selbst einstellende maximale Spannung U_{Ph} der Photozelle bei Bestrahlung mit Licht der Wellenlängen $\lambda_{\mathrm{CWL}}^{(i)}$ bei maximaler Lichtintensivität auf.
- Verwenden Sie die folgenden sechs Wellenlängen der zur Verfügung stehenden Filter: $\lambda_{\mathrm{CWL}}^{(i)} =$ 360, 400, 440, 490, 540, 590 nm.
- Bestimmen Sie h durch Anpassung eines geeigneten Modells.

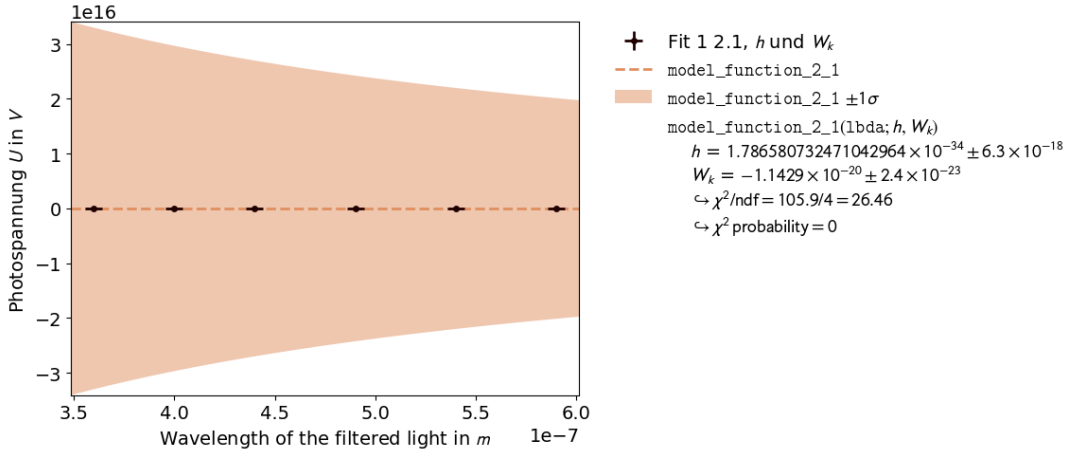
For this task, we made four series of measurement. Because we can fit a model function only on one series of measurement, we need to take the weighted average of the values and their standard deviation, but because all values have the same standard deviation, we can just take the normal average. With the following formulas, we get the weighted average and its standard deviation:
- $\bar{n} = \frac{\sum w_i n_i}{\sum w_i} =$, where $w_i = \frac{1}{\Delta n_i}$
- $\sigma_{\bar{n}} = \sqrt{\frac{N}{N-1} \frac{\sum w_i (n_i - \bar{n})^2}{\sum w_i}}$

The light after passing the various Fabry-Perot-Color-Filters has a certain *central wavelength* with a specified **FWHM** of $\pm 10\,nm$. To convert this to a standard deviation we can use the following conversion: $FWHM = 2\sqrt{2\ln(2)}\sigma \Leftrightarrow \sigma = \frac{FWHM}{2\sqrt{2\ln(2)}} = \frac{\pm 10\,nm}{2\sqrt{2\ln(2)}} \approx \pm 4.25\,nm$ because in the preparation material it is given that a normal distribution can be assumed.

[ ]:
```
U_1 = np.array([0.794,0.765,0.708,0.633,0.569,0.472])
U_2 = np.array([0.793,0.765,0.709,0.633,0.568,0.473])
U_3 = np.array([0.793,0.764,0.709,0.634,0.569,0.472])
U_4 = np.array([0.794,0.765,0.709,0.634,0.570,0.472])
lam = np.array([360,400,440,490,540,590]) *10**(-9)

n1 = np.array([ U_1[0] , U_2[0] , U_3[0] , U_4[0]])
n2 = np.array([ U_1[1] , U_2[1] , U_3[1] , U_4[1]])
n3 = np.array([ U_1[2] , U_2[2] , U_3[2] , U_4[2]])
n4 = np.array([ U_1[3] , U_2[3] , U_3[3] , U_4[3]])
n5 = np.array([ U_1[4] , U_2[4] , U_3[4] , U_4[4]])
n6 = np.array([ U_1[5] , U_2[5] , U_3[5] , U_4[5]])

U_bar = np.array([ np.sum(n1) , np.sum(n2) , np.sum(n3) , np.sum(n4) , np.
 ↪sum(n5) , np.sum(n6) ]) / 4
std_U_bar = np.sqrt( np.array([ np.sum( (n1 - U_bar[0])**2 ) , np.sum( (n2 -␣
 ↪U_bar[1])**2 ) , np.sum( (n3 - U_bar[2])**2 ) , np.sum( (n4 - U_bar[3])**2 )␣
 ↪, np.sum( (n5 - U_bar[4])**2 ) , np.sum( (n6 - U_bar[5])**2 ) ]) / 3)
```

```
std_lam = 10**(-8) / ( 2 * np.sqrt(2 * np.log(2)))
```

[ ]:
```
def model_function_2_1(lbda, h=10**(-34), W_k=0):
    return (c*h/e) * (1/lbda) + (W_k/e)

data_2_1 = np.array([ lam , U_bar ])
error_2_1 = np.array([ np.ones(lam.size)*std_lam , std_U_bar ])
label_2_1 = [ "Wavelength of the filtered light in $m$" , "Photospannung $U$ in␣
 ↪$V$" ]
title_2_1 = "Fit 1 2.1, $h$ und $W_k$"

fit_2_1_res = fit_funktion(data_2_1, model_function_2_1, error_2_1, label_2_1,␣
 ↪title_2_1)
fit_2_1_res[2].plot()
fit_2_1_res[2].show()

print(f"Plancks constant in Js:", fit_2_1_res[0][0], "Js")
print(f"Work function of K in eV:", fit_2_1_res[0][1]/e, "eV")
print(f"Std of the Work function of K in eV", fit_2_1_res[1][1]/e, "eV")
```



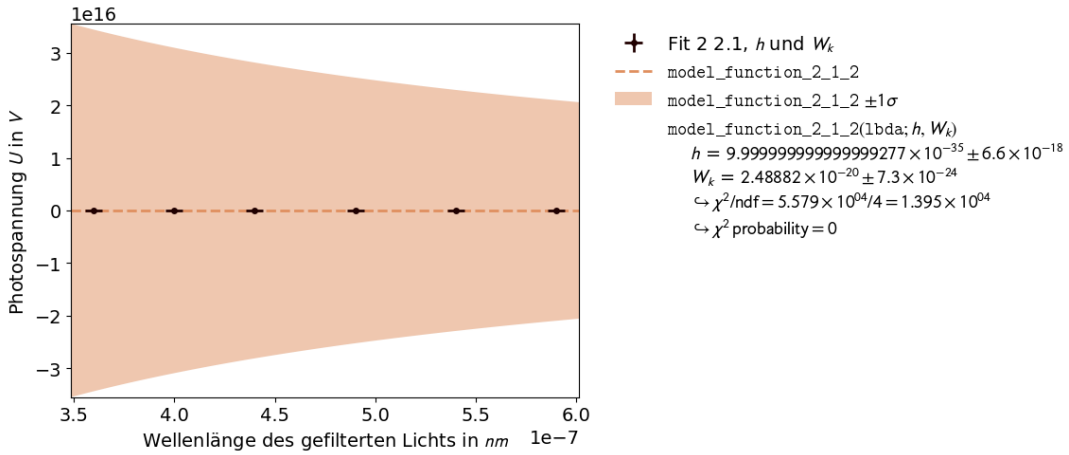

```
Plancks constant in Js: 1.786580732471043e-34 Js
Work function of K in eV: -0.07133648091461918 eV
Std of the Work function of K in eV 0.0001502202050864155 eV
```

Unforntunately, the fit doesn't really work out. It returns the value for Planck's constant as $(1.79 \cdot 10^{-34} \pm 6.3 \cdot 10^{-18})\, Js$ which is the same order of magnitude as the real value, but the χ^2 probability is just 0. I really don't know where the problem is. The most plausible explanation is, of course, human error, but since this experiment was pretty straightforward, I don't know where we could have gone wrong. However, after consulting with other groups which have already completed this experiment, we found that our measurements were indeed off by a factor of 2 for the highest photon energies and a factor of 1/2 for the lowest.
On the other hand, the fit yields a value for the work function of $(-0.0713 \pm 0.0002)\, eV$. This is

about 2 orders of magnitude off the real value for potassium (K) of 2.3 eV (Source), but since the fit isn't anywhere near being good, it's useless to talk about this beyond mentioning it, anyway.

Now, if we repeat the fit with the other groups measurement, we get the following:

[ ]:
```
def model_function_2_1_2(lbda, h=10**(-34), W_k=0):
    return (c*h/e) * (1/lbda) + (W_k/e)

U_1_ = np.array([1.618,1.1747,0.865,0.5873,0.3920,0.1983])
U_2_ = np.array([1.619,1.1746,0.864,0.5872,0.3926,0.1978])
U_3_ = np.array([1.6199,1.1745,0.8622,0.5874,0.3919,0.1982])

n1_ = np.array([ U_1_[0] , U_2_[0] , U_3_[0]])
n2_ = np.array([ U_1_[1] , U_2_[1] , U_3_[1]])
n3_ = np.array([ U_1_[2] , U_2_[2] , U_3_[2]])
n4_ = np.array([ U_1_[3] , U_2_[3] , U_3_[3]])
n5_ = np.array([ U_1_[4] , U_2_[4] , U_3_[4]])
n6_ = np.array([ U_1_[5] , U_2_[5] , U_3_[5]])

U_bar_ = np.array([ np.sum(n1_) , np.sum(n2_) , np.sum(n3_) , np.sum(n4_) , np.
 ↪sum(n5_) , np.sum(n6_) ]) / 3
std_U_bar_ = np.sqrt( np.array([ np.sum( (n1_ - U_bar_[0])**2 ) , np.sum( (n2_␣
 ↪- U_bar_[1])**2 ) , np.sum( (n3_ - U_bar_[2])**2 ) , np.sum( (n4_ -␣
 ↪U_bar_[3])**2 ) , np.sum( (n5_ - U_bar_[4])**2 ) , np.sum( (n6_ -␣
 ↪U_bar_[5])**2 ) ]) / 2)

data_2_1_2 = np.array([ lam , U_bar_ ])
error_2_1_2 = np.array([ np.ones(lam.size)*std_lam , std_U_bar_ ])
label_2_1_2 = [ "Wellenlänge des gefilterten Lichts in $nm$" , "Photospannung␣
 ↪$U$ in $V$" ]
title_2_1_2 = "Fit 2 2.1, $h$ und $W_k$"

fit_2_1_res2 = fit_funktion(data_2_1_2, model_function_2_1_2, error_2_1_2,␣
 ↪label_2_1_2, title_2_1_2)
fit_2_1_res2[2].plot()
fit_2_1_res2[2].show()

print(f"Plancks constant in Js:", fit_2_1_res2[0][0], "Js")
print(f"Work function of K in eV:", fit_2_1_res2[0][1]/e, "eV")
print(f"Std of the Work function of K in eV", fit_2_1_res2[1][1]/e, "eV")
```




```
Plancks constant in Js: 1e-34 Js
Work function of K in eV: 0.1553396319220421 eV
Std of the Work function of K in eV 4.552092835431482e-05 eV
```

Unfortunately again, the fit doesn't yield the correct value for Plancks constant, as before. This time though, it's about $(10^{-34} \pm 6.6 \cdot 10^{-18})\,Js$. How can two completely uncorrelated series of measurement yield a value so wrong in the same way? The value for the work function is $(0.15534 \pm 4.6e-05)\,eV$, which is not even negative. After many hours of contemplation, I'm out of ideas as to why this would happen.

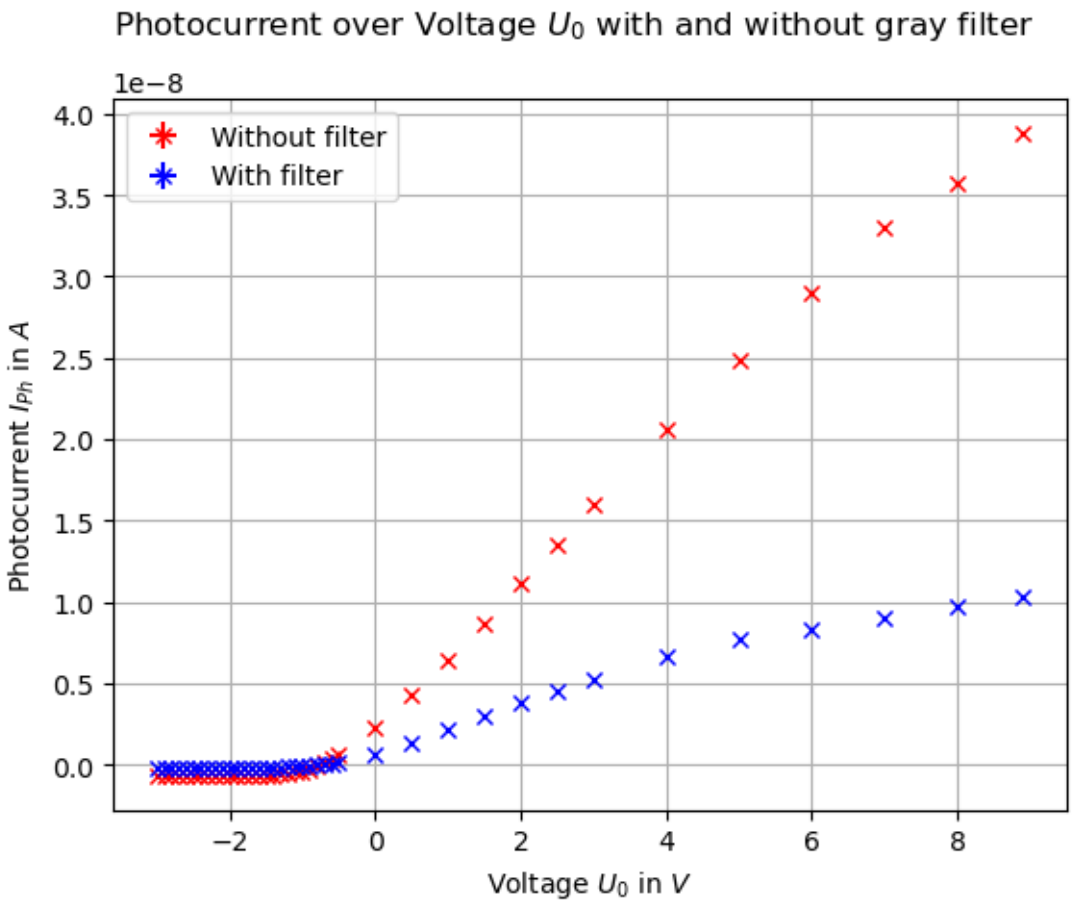
Now, for the evaluation and analysis of this part of the experiment, I did follow the instructions closely so I don't really understand why the fits would yield results so wrong, even with a second set of measurements which yielded much better results for the other group.

2.2.2 Aufgabe 2.2: Photostrom I_{Ph} als Funktion einer angelegten externen Spannung U_o bei variierender Lichtintensität

- Tragen Sie für $\lambda_{\mathrm{CWL}} = 400\,\mathrm{nm}$ den Photostrom I_{Ph} als Funktion einer angelegten externen Spannung U_o auf.
- Bestimmen Sie durch Anpassung eines geeigneten Modells den Wert von $U(I_{\mathrm{Ph}} = 0)$ im Nahbereich des Nulldurchgangs.
- Folgen Sie den vorgenannten Punkten einmal für die maximale und einmal für eine bei mittels eines Graufilters reduzierte Lichtintensität.
- Bestimmen Sie die Abnahme der Lichtintensität durch den Filter.

Here, we plot the photocurrent over the voltage U_E with and without the gray filter. For the photocurrent we use the following formula: $I_{Ph} = \frac{U_E}{R}$, where the amplification has already been absorbed into the measurement of U_E.

[ ]:
```
# in V, without gray filter
U_e_high = np.array([-0.068,-0.068,-0.068,-0.068,-0.068,-0.068,-0.068,-0.068,-0.
 ↪068,-0.068,-0.068,-0.067,-0.067,-0.067,
                     -0.066,-0.065,-0.064,-0.061,-0.057,-0.049,-0.039,-0.026,-0.
 ↪008,0.0113,0.034,0.061,
                     0.225,0.432,0.639,0.866,1.111,1.353,1.597,
                     2.060,2.490,2.905,3.300,3.570,3.883])
# in V, with gray filter
U_e_low = np.array([-0.014,-0.014,-0.014,-0.014,-0.014,-0.014,-0.014,-0.014,-0.
 ↪014,-0.014,-0.014,-0.014,-0.014,-0.014,
                     -0.014,-0.014,-0.014,-0.014,-0.013,-0.011,-0.009,-0.006,-0.
 ↪002,0.002,0.008,0.015,
                     0.067,0.133,0.213,0.300,0.383,0.457,0.526,
                     0.660,0.770,0.837,0.901,0.971,1.030]) #in V
U_0 = np.array([-3,-2.9,-2.8,-2.7,-2.6,-2.5,-2.4,-2.3,-2.2,-2.1,-2,-1.9,-1.8,-1.
 ↪7,-1.6,-1.5,-1.4,-1.3,-1.2,-1.1,-1.,-0.9,-0.8,-0.7,-0.6,-0.5,
                     0.,0.5,1.,1.5,2.,2.5,3,
                     4.,5.,6.,7.,8.,8.89]) #in V
U_std = 0.01
R = 100 * 10**6


fig, ax = plt.subplots()
ax.errorbar(U_0, U_e_high/R, xerr=U_std, yerr=U_std/R, fmt="rx", label="Without␣
 ↪filter")
ax.errorbar(U_0, U_e_low/R, xerr=U_std, yerr=U_std/R, fmt="bx", label="With␣
 ↪filter")
fig.suptitle("Photocurrent over Voltage $U_0$ with and without gray filter")
ax.set(xlabel="Voltage $U_0$ in $V$", ylabel="Photocurrent $I_{Ph}$ in $A$")
ax.grid(), ax.legend()
plt.show()
```



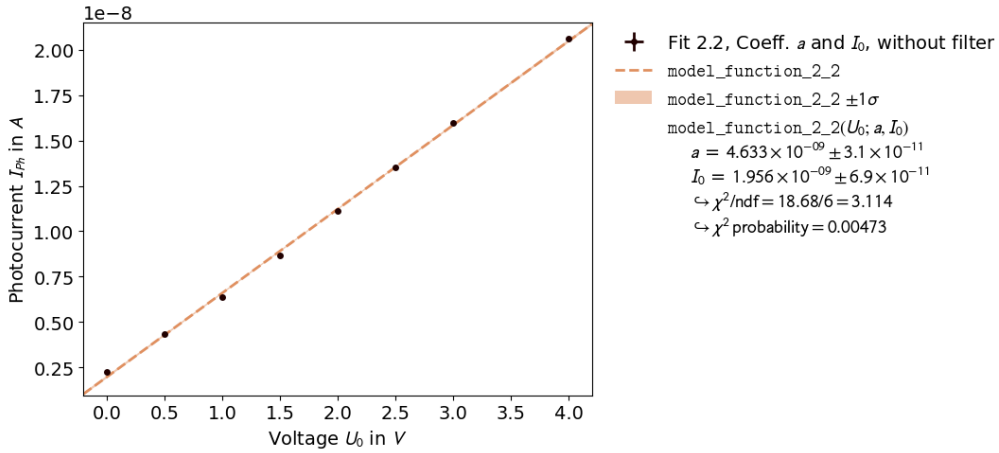
As the blue graph is only linear between $U_0 = 0\,V$ and $U_0 = 4\,V$, we'll only use those values for both fits. The model function will be

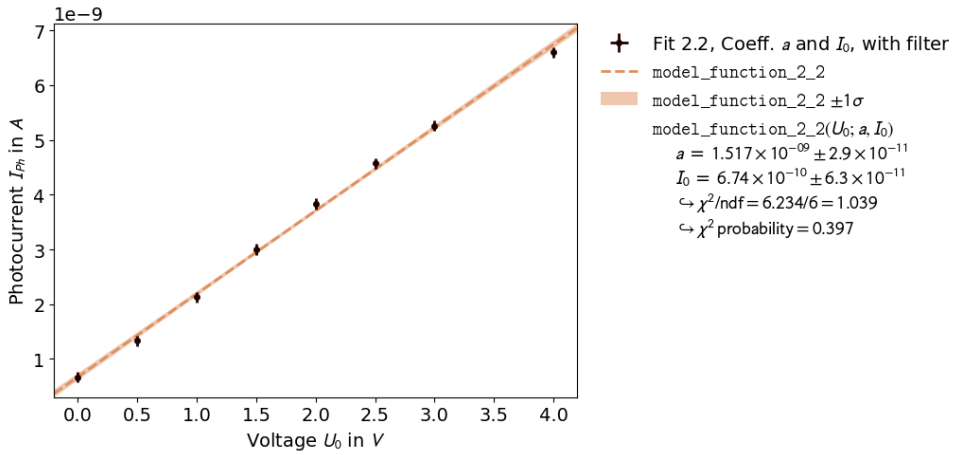
$$I_{Ph} = a \cdot U_0 + I_0$$

, where $[a] = 1/\Omega$ and $[I_0] = A$

[ ]:
```
def model_function_2_2(U_0, a=10**(-9), I_0=10**(-10)):
    return a * U_0 + I_0

data_2_2_high = np.array([ U_0[26:34] , U_e_high[26:34]/R ])
data_2_2_low = np.array([ U_0[26:34] , U_e_low[26:34]/R ])
error_2_2 = np.array([ U_std , U_std/R ])
label_2_2 = [ "Voltage $U_0$ in $V$" , "Photocurrent $I_{Ph}$ in $A$" ]
title_2_2_high = "Fit 2.2, Coeff. $a$ and $I_0$, without filter"
title_2_2_low = "Fit 2.2, Coeff. $a$ and $I_0$, with filter"

fit_2_2_high_res = fit_funktion(data_2_2_high, model_function_2_2, error_2_2,␣
↪label_2_2, title_2_2_high)
```

```
fit_2_2_low_res = fit_funktion(data_2_2_low, model_function_2_2, error_2_2,␣
 ↪label_2_2, title_2_2_low)

fit_2_2_high_res[2].plot()
fit_2_2_low_res[2].plot()
fit_2_2_high_res[2].show()
fit_2_2_low_res[2].show()

print(f"Coefficients a, without and with filter, respectively:")
print(f"a=(4.633 +/- .031) * 10^(-9) 1/Ohm")
print(f"a=(1.517 +/- .029) * 10^(-9) 1/Ohm\n")
print(f"Coefficients $I_0$, without and with filter, respectively:")
print(f"I0=(1.956 +/- .069) * 10^(-9) Ampere")
print(f"a=(0.674 +/- .063) * 10^(-9) Ampere")
```






```
Coefficients a, without and with filter, respectively:
```

```
a=(4.633 +/- .031) * 10^(-9) 1/Ohm
a=(1.517 +/- .029) * 10^(-9) 1/Ohm

Coefficients $I_0$, without and with filter, respectively:
I0=(1.956 +/- .069) * 10^(-9) Ampere
a=(0.674 +/- .063) * 10^(-9) Ampere
```

For once, our fits do yield nice results. While the χ^2 probability of the fit without filter is very low, the second fit's probability is very good. The reason the first is so low, though, is that the errors of the measurements are the same for both fits, but the actual values of the first fit are larger by one order of magnitude, so the relative errors turn out much lower and thus the fit doesn't have much space for adjusting the parameters. So, the results and goodness of fit:
- Without filter are:
- $a = (4.633 \pm 0.031)\,\frac{1}{G\Omega}$
- $I_0 = (1.956 \pm 0.069)\,nA$
- $\chi^2 probability = 0.00473$
- With filter are:
- $a = (1.517 \pm 0.029)\,\frac{1}{G\Omega}$
- $I_0 = (0.674 \pm 0.063)\,nA$
- $\chi^2 probability = 0.397$

Using these results and the model function we started with we can calculate the voltage U_0 for which the Photocurrent falls to 0. Equating I_{Ph} with 0 and rearranging the formula, we get $U_0(I_{Ph} = 0) = -\frac{I_0}{a}$. The resulting error on $U_0(I_{Ph})$ we can get from propagation of uncertainty.

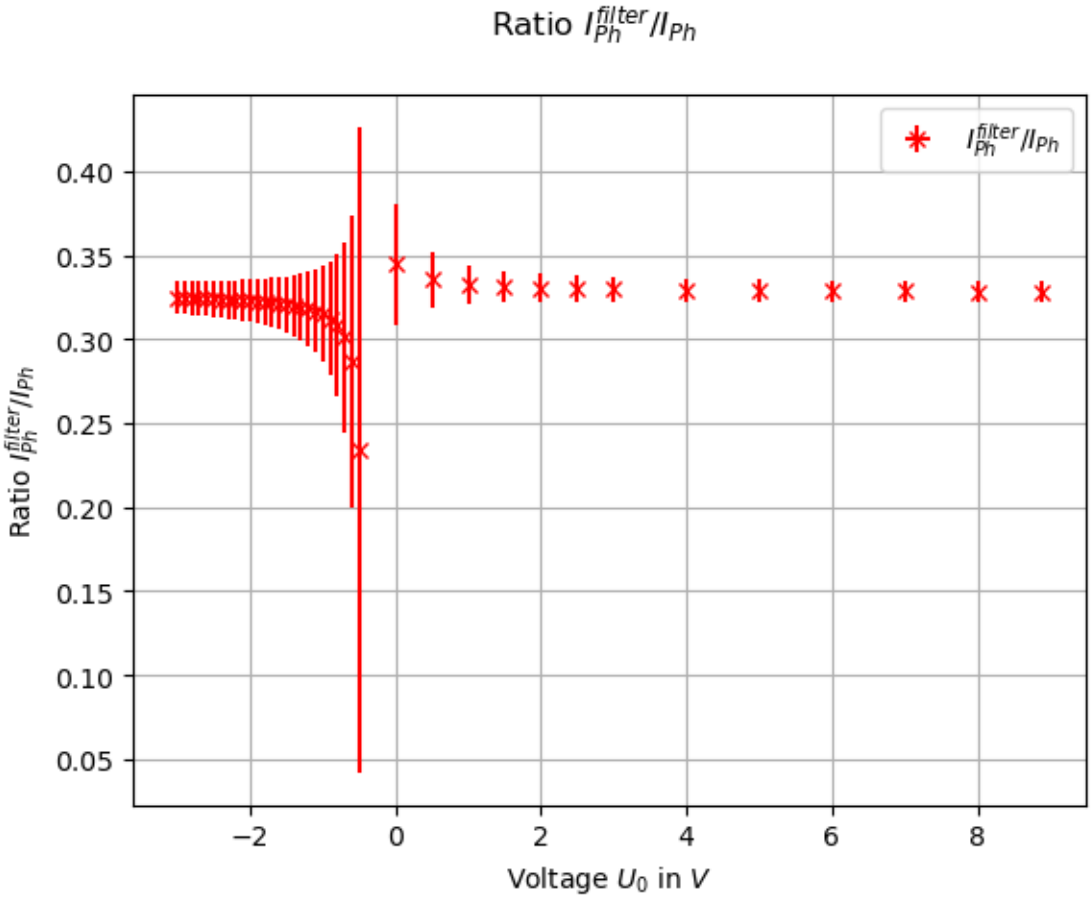
[ ]:
```
# Without filter
a = fit_2_2_high_res[0][0]
a_std = fit_2_2_high_res[1][0]
I_0 = fit_2_2_high_res[0][1]
I_0_std = fit_2_2_high_res[1][1]
U_0_I_Ph0 = - I_0 / a
U_0_I_Ph0_std = np.sqrt( I_0_std**2 + (I_0**2 * (a_std**2) / a**2) ) / a

# With filter
a_filter = fit_2_2_low_res[0][0]
a_filter_std = fit_2_2_low_res[1][0]
I_0_filter = fit_2_2_low_res[0][1]
I_0_filter_std = fit_2_2_low_res[1][1]
U_0_I_Ph0_filter = - I_0_filter / a_filter
U_0_I_Ph0_filter_std = np.sqrt( I_0_filter_std**2 + (I_0_filter**2 *␣
 ↪(a_filter_std**2) / a_filter**2) ) / a_filter

print(f"Without filter: U_0(I_Ph=0)=({U_0_I_Ph0:.3f} +/- {U_0_I_Ph0_std:.3f})V")
print(f"With filter: U_0(I_Ph=0)=({U_0_I_Ph0_filter:.3f} +/-␣
 ↪{U_0_I_Ph0_filter_std:.3f})V")
```

```
Without filter: U_0(I_Ph=0)=(-0.422 +/- 0.015)V
With filter: U_0(I_Ph=0)=(-0.444 +/- 0.042)V
```

So, the voltage that completely stops the photocurrent without and with filter is:
- $U_0(I_{Ph} = 0) = (-0.422 \pm 0.015)\,V$ without filter
- $U_0(I_{Ph} = 0) = (-0.444 \pm 0.042)\,V$ with filter

Lastly, we plot the ratio I_{Ph}^{Filter}/I_{Ph}:

[ ]:
```
I_Ph = a * U_0 + I_0
I_Ph_std = np.sqrt( (U_0 * a_std)**2 + (a * U_std)**2 + (I_0_std)**2 )
I_Ph_filter = a_filter * U_0 + I_0_filter
I_Ph_filter_std = np.sqrt( (U_0 * a_filter_std)**2 + (a_filter * U_std)**2 +␣
 ↪(I_0_filter_std)**2 )

fig2, ax2 = plt.subplots()
ax2.errorbar(U_0, I_Ph_filter/I_Ph, xerr=U_std, yerr= np.sqrt(␣
 ↪I_Ph_filter_std**2 + (I_Ph_filter*I_Ph_std/I_Ph)**2 )/np.absolute(I_Ph),␣
 ↪fmt="rx", label="$I_{Ph}^{filter}/I_{Ph}$")
fig2.suptitle("Ratio $I_{Ph}^{filter}/I_{Ph}$")
ax2.set(xlabel="Voltage $U_0$ in $V$", ylabel="Ratio $I_{Ph}^{filter}/I_{Ph}$")
ax2.grid(), ax2.legend()
plt.show()
```



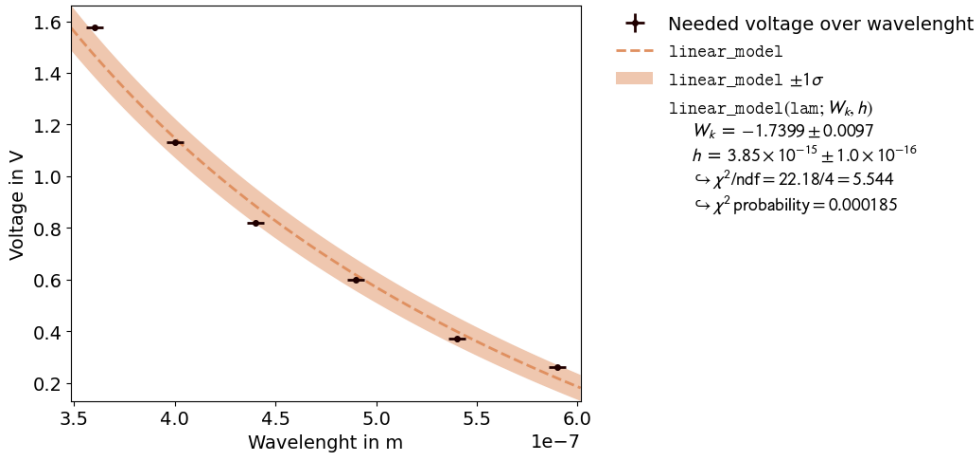
In this plot we can nicely see how, for most voltages, the ratio is independent of the voltage and, more importantly, the filtered light has an intensity of about a third of the unfiltered light, because for voltages lower than about $-2\,V$ and higher than about $1\,V$ the graph is horizontal. Only when the value of $U_0(I_{Ph} = 0)$ is approached, the ratio diverges to negative infinity from the left and positive infinity from the right. So the filter reduces the intensity of the light by about two thirds.

2.2.3 Aufgabe 2.3: Spannung $U_o(I_{\mathrm{Ph}} = 0)$ bei variierender Lichtfrequenz

- Bestimmen Sie für die in **Aufgabe 2.1** verwendeten Wellenlängen $\lambda_{\mathrm{CWL}}^{(i)}$ jeweils die Spannung $U^{(i)}(I_{Ph} = 0)$.
- Bestimmen Sie h durch Anpassung eines geeigneten Modells und vergleichen Sie mit dem Ergebnis aus **Aufgabe 2.1**.

In this part of the experiment, we again want to obtain a value for h. This is done by varying U_0 until it exactly opposes U_{ph} and then fitting the model $U_0 = \frac{ch}{\lambda_{cwl}} + W_k$. The uncertainties for the wavelengths λ are the same as in part 2.1.

[ ]:
```
def linear_model(lam,W_k=1,h=10**-15):
    return c/lam*h + W_k
U_0_2_3 = np.array([1.575, 1.13, 0.82, 0.60, 0.37, 0.26]) # in V

h = 6.62607015 * 10**(-34)

data_2_3 = np.array([ lam , U_0_2_3])
error_2_3 = np.array([ std_lam , U_std])
label_2_3 = [ "Wavelenght in m" , "Voltage in V" ]
title_2_3 = "Needed voltage over wavelenght"
fit_2_3_res = fit_funktion(data_2_3, linear_model, error_2_3, label_2_3,␣
 ↪title_2_3)
fit_2_3_res[2].plot()
fit_2_3_res[2].show()

print(f"Value for Plancks constant:({fit_2_3_res[0][1]} +/-␣
 ↪{fit_2_3_res[1][1]})eVs")
print(f"Value for Plancks constant:({fit_2_3_res[0][1]*e} +/-␣
 ↪{fit_2_3_res[1][1]*e})Js")
print(f"Ratio of h from the fit and the real value of h: {fit_2_3_res[0][1]*e /␣
 ↪h}")
```




```
Value for Plancks constant:(3.8522275751213784e-15 +/-
1.0154334538835234e-16)eVs
Value for Plancks constant:(6.171949009709953e-34 +/- 1.626903753194098e-35)Js
Ratio of h from the fit and the real value of h: 0.9314644834706366
```

While the fit is not too great, as can be seen from the χ^2 probability, it yields a value for Plancks constant that's not too far off from the actual value:

$$h = (3.852 \pm 0.102)10^{-15}\, eVs$$

or alternatively in SI units:

$$h = (6.172 \pm 0.163)10^{-34}\, Js$$

The exact value:

$$h = 6.62607015 \cdot 10^{-34}\, Js$$

Unforntuately, a comparison with the value obtained in 2.1 is not really possible, because, as already discussed in 2.1, neither fit yielded useful results. But a comparison to the exact, known value can be made. While the real value does not lie within the uncertainty interval of our value, their ratio is $\frac{h_{Fit}}{h_{exp}} = \frac{6.17194901}{6.62607015} \approx 0.931 = 93.1\%$, so that is not entirely bad.