

PW6

November 4, 2025

```
[65]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import Latex, HTML, Math, display

from uncertainties import ufloat
from uncertainties.umath import sqrt
from uncertainties import unumpy as unp
import matplotlib.pyplot as plt
import numpy as np
from uncertainties.umath import *
import pandas as pd
from uncertainties.umath import sin, radians #achtung in bogenmaß eingeben
from IPython.display import display, Latex

import numpy as np #numpy ist ein package mit allen mathematischen ausdrücken/
↪funktionen/operationen/etc.
from uncertainties import ufloat #Uncertainties ist für das automatische
↪berechnen von unsicherheiten
from uncertainties.umath import * #Importieren von rechnen mit uncertainties
↪(addition/multiplikation/etc.)
from uncertainties import unumpy as unp #kann arrays von unsicherheiten
↪verstehen (also eine liste, statt jede zahl einzeln einzutippen)
from IPython.display import Latex, HTML, Math, display # Um ergebnisse schöner
↪anzuzeigen. display() "ersetzt" quasi print(), und lässt latex fromatierung
↪zu
import matplotlib.pyplot as plt #Damit kann man graphen plotten
```

1 Strom und Spannungsquellen

07.11.2025 Anna Andreatta, Jeppe Vogler

1.1 Solarzellen als Gleichstromquelle

1.1.1 Grundlagen

Solarzellen beruhen auf dem inneren Photoeffekt. Im Grundzustand gibt es ein mit Elektronen vollbesetztes Valenzband und ein leeres Leitungsband. Diese haben eine Energielücke von etwa 1 eV.

Die Elektronen müssen diese Lücke überwinden, damit der Halbleiter elektrisch leitend wird. Das passiert beim inneren Photoeffekt durch optische Anregung. Wenn Photonen mit einer Energie größer als der Energielücke auf den Halbleiter treffen, werden die Elektronen vom Valenzband ins Leitungsband angeregt, in dem sie sich frei bewegen können. Da die Anzahl der angeregten Ladungsträger gleich der Anzahl der absorbierten Photonen ist, hängt die Anzahl der erzeugten Ladungsträger von der Intensität und Wellenlänge des Lichts ab.

```
[112]: Unsicherheit_Strom = 0.0003
        Unsicherheit_Klemmspannung = 0.017

        Klemmspannung1_input = "0.590, 0.831, 1.092, 1.310, 1.545, 1.940, 2.239, 2.750,
        ↪2.866, 2.915, 2.957, 2.967, 2.986" #in V
        Strom1_input = "0.0116, 0.0116, 0.0116, 0.0115, 0.0115, 0.0111, 0.0102, 0.0064,
        ↪0.0045, 0.0034, 0.0023, 0.0020, 0.0014" #in A

        Klemmspannung1 = np.array([float(v.strip())for v in Klemmspannung1_input.
        ↪split(",")])
        Strom1 = np.array([float(i.strip()) for i in Strom1_input.split(",")])

        Klemmspannung2_input = "3.258, 3.248, 3.243, 3.236, 3.222, 3.197, 3.144, 3.095,
        ↪2.968, 2.850, 2.219, 1.314, 0.516" #in V
        Strom2_input = "0.0015, 0.0019, 0.0021, 0.0026, 0.0034, 0.0054, 0.010, 0.0129,
        ↪0.0161, 0.0171, 0.0175, 0.0176, 0.0177" #in A

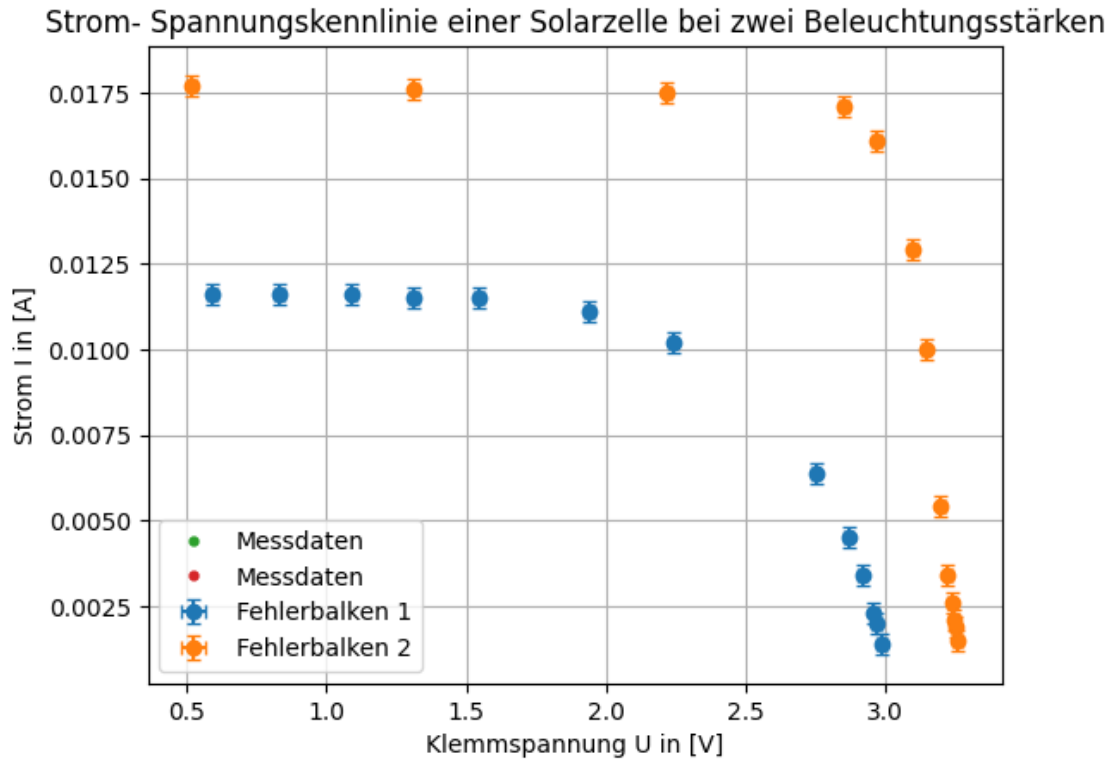
        Klemmspannung2 = np.array([float(v.strip())for v in Klemmspannung2_input.
        ↪split(",")])
        Strom2 = np.array([float(i.strip()) for i in Strom2_input.split(",")])

        plt.figure()
        plt.errorbar(Klemmspannung1, Strom1, xerr=Unsicherheit_Klemmspannung,
        ↪yerr=Unsicherheit_Strom, fmt='o', capsize=3, label="Fehlerbalken 1",
        ↪zorder=10);
        plt.errorbar(Klemmspannung2, Strom2, xerr=Unsicherheit_Klemmspannung,
        ↪yerr=Unsicherheit_Strom, fmt='o', capsize=3, label="Fehlerbalken 2",
        ↪zorder=10);
        plt.plot(Klemmspannung1, Strom1, 'o', markersize=3.5, label="Messdaten",
        ↪zorder=10)
        plt.plot(Klemmspannung2, Strom2, 'o', markersize=3.5, label="Messdaten",
        ↪zorder=10)
        plt.xlabel("Klemmspannung U in [V]")
        plt.ylabel("Strom I in [A]")
        plt.title("Strom- Spannungskennlinie einer Solarzelle bei zwei
        ↪Beleuchtungsstärken")
        plt.legend()
        plt.grid(True)
        plt.show()
```

```

print("I_KS1 = (0,0116 +/- 0.0003) A")
print("I_KS2 = (0,0177 +/- 0,0003) A")
print("U_LL1 = (3,0 +/- 0,1) V")
print("U_LL2 = (3,3 +/- 0,1) V")

```



```

I_KS1 = (0,0116 +/- 0.0003) A
I_KS2 = (0,0177 +/- 0,0003) A
U_LL1 = (3,0 +/- 0,1) V
U_LL2 = (3,3 +/- 0,1) V

```

```

[108]: Unsicherheit_Strom = 0.0003
        Unsicherheit_Klemmspannung = 0.017
        Klemmspannung1 = unp.uarray([0.590 , 0.831 , 1.092 , 1.310 , 1.545 , 1.940 , 2.
        ↪239 , 2.750 ,2.866 , 2.915 , 2.957 , 2.967 , 2.986],␣
        ↪Unsicherheit_Klemmspannung)
        Strom1 = unp.uarray([0.0116 , 0.0116 , 0.0116 , 0.0115 , 0.0115 , 0.0111 , 0.
        ↪0102 , 0.0064 , 0.0045 , 0.0034 , 0.0023 , 0.0020 , 0.0014],␣
        ↪Unsicherheit_Strom)

        Klemmspannung2 = unp.uarray([3.258 , 3.248 , 3.243 , 3.236 , 3.222 , 3.197 , 3.
        ↪144 , 3.095 , 2.968 , 2.850 , 2.219 , 1.314 , 0.516],␣
        ↪Unsicherheit_Klemmspannung)

```

```

Strom2 = unp.uarray([0.0015 , 0.0019 , 0.0021 , 0.0026 , 0.0034 , 0.0054 , 0.
↳0100 , 0.0129 , 0.0161 , 0.0171 , 0.0175 , 0.0176 , 0.0177],↳
↳Unsicherheit_Strom)

leistung1 = Klemmspannung1 * Strom1
leistung2 = Klemmspannung2 * Strom2

Lastenwiderstand1 = Klemmspannung1/Strom1
Lastenwiderstand2 = Klemmspannung2/Strom2

print("Leistung1:", leistung1)
print("Leistung2:", leistung2)

print("Maximaler Widerstand1:", Lastenwiderstand1)
print("Maximaler Widerstand:", Lastenwiderstand2)

R1_nom = unp.nominal_values(Lastenwiderstand1)
R1_std = unp.std_devs(Lastenwiderstand1)

R2_nom = unp.nominal_values(Lastenwiderstand2)
R2_std = unp.std_devs(Lastenwiderstand2)

P1_nom = unp.nominal_values(leistung1)
P1_std = unp.std_devs(leistung1)

P2_nom = unp.nominal_values(leistung2)
P2_std = unp.std_devs(leistung2)

plt.figure()
plt.plot(R1_nom, P1_nom, 'o', markersize=3.5, label="Leistung1", zorder=10)
plt.plot(R2_nom, P2_nom, 'o', markersize=3.5, label="Leistung2", zorder=10)
plt.xlabel("Lastenwiderstand in [Ohm]")
plt.ylabel("Leistung in [W]")
plt.title("Leistung")
plt.legend()
plt.grid(True)
plt.show()

print("P_max1 = (0,0228 +/- 0,0011) W")
print("R_Lmax = (220 +/- 2) Ohm")
print("P_max2 = (0.0487 +/- 0,0015) W")
print("R_Lmax2 = (167 +/-1) Ohm")

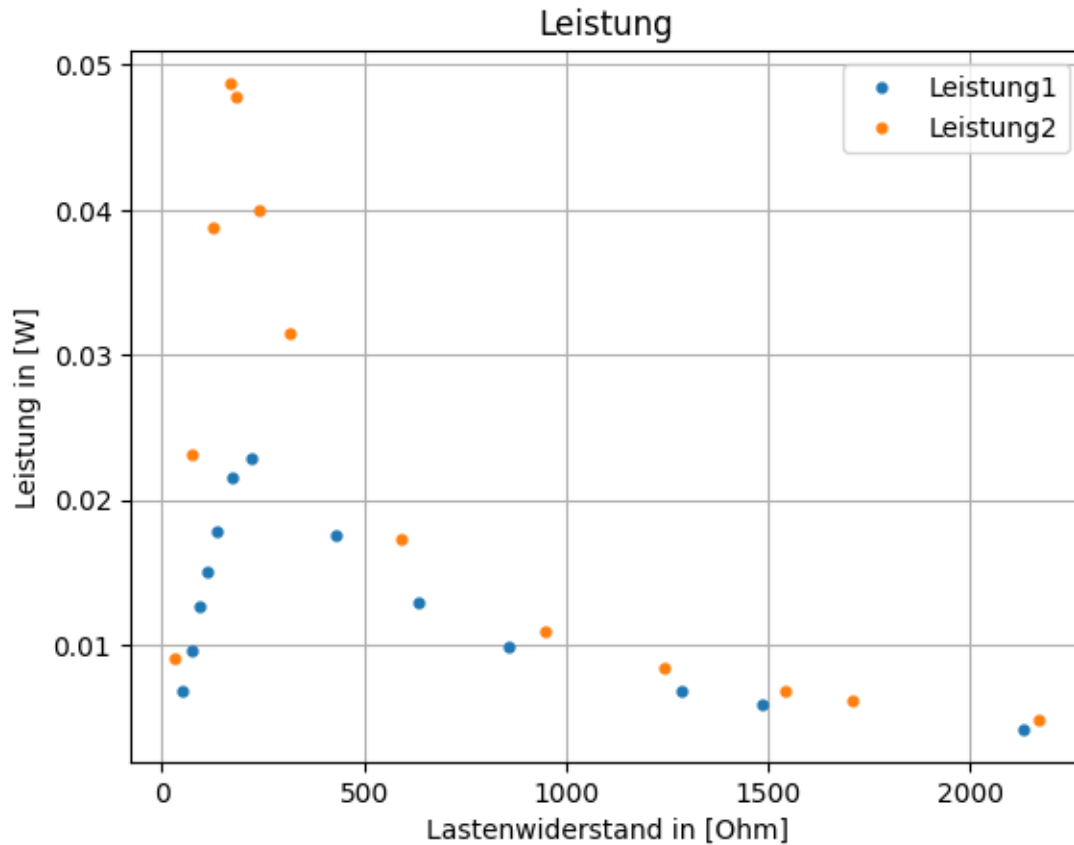
```

```

Leistung1: [0.006843999999999999+/-0.00026498460332630646
0.009639599999999998+/-0.0003178652702010712
0.0126672+/-0.0003823736392587753 0.015065+/-0.0004389410552682444
0.0177675+/-0.0005030432386982256 0.021534+/-0.0006118265195298419

```

0.022837800000000002+/-0.0006937207291122271
 0.0176+/-0.0008321432809318356
 0.012896999999999999+/-0.0008631965535148991
 0.009911+/-0.0008764080613504191
 0.006801099999999999+/-0.0008879612716779937
 0.005934+/-0.000890749128542936 0.0041804+/-0.0008961161085484403]
 Leistung2: [0.004887+/-0.0009777325861399936
 0.006171200000000001+/-0.000974935202975049
 0.006810299999999999+/-0.0009735547750383641
 0.0084136+/-0.0009718056801645069
 0.010954799999999999+/-0.0009683265977964251
 0.017263800000000003+/-0.0009634832899433181
 0.03144+/-0.0009583977462410896 0.0399255+/-0.0009540465083003028
 0.047784799999999995+/-0.000931516961734997
 0.048735+/-0.0009030678213733451 0.0388325+/-0.0007291520691872169
 0.023126400000000002+/-0.0004948881489791405
 0.009133200000000001+/-0.0003383841751619009]
 Maximaler Widerstand1: [50.862068965517246+/-1.9692672660991861
 71.63793103448276+/-2.3622567642766885
 94.13793103448278+/-2.8416590313523726
 113.91304347826087+/-3.3190249925765176
 134.34782608695653+/-3.80372959318129
 174.77477477477476+/-4.965721285040514
 219.5098039215686+/-6.667827077203257 429.6875+/-20.315998069624893
 636.8888888888889+/-42.626990297032066
 857.3529411764706+/-75.8138461375795
 1285.6521739130435+/-167.85657309602905 1483.5+/-222.687282135734
 2132.857142857143+/-457.2020961981839]
 Maximaler Widerstand: [2172.0+/-434.54781606221934
 1709.4736842105265+/-270.0651531786839
 1544.2857142857142+/-220.7607199633479
 1244.6153846153848+/-143.7582367107259
 947.6470588235295+/-83.76527662598835
 592.0370370370371+/-33.04126508721941
 314.40000000000003+/-9.583977462410894
 239.92248062015506+/-5.733108036177531
 184.34782608695653+/-3.593676793854392
 166.66666666666666+/-3.0883616202364665
 126.79999999999998+/-2.3809047157133607
 74.6590909090909+/-1.597650274338651
 29.152542372881356+/-1.0800988705732735]



```
P_max1 = (0,0228 +/- 0,0011) W
R_Lmax = (220 +/- 2) Ohm
P_max2 = (0.0487 +/- 0,0015) W
R_Lmax2 = (167 +/-1) Ohm
```

```
[113]: IKS_1 = ufloat(0.0116, 0.0003)
       UKK_1 = ufloat(3, 0.1)
       P_max1 = ufloat(0.0228, 0.0011)

       CFF1 = P_max1 / (IKS_1 * UKK_1)

       IKS_2 = ufloat(0.0177, 0.0003)
       UKK_2 = ufloat(3.3, 0.1)
       P_max2 = ufloat(0.0487, 0.0015)

       CFF2 = P_max2 / (IKS_2 * UKK_2)

       print(f"CFF1 = {CFF1:.3uP}")
       print(f"CFF2 = {CFF2:.3uP}")
```

```
CFF1 = 0.6552±0.0420
```

CFF2 = 0.8338±0.0387

1.2 Reale Spannungsquelle

1.2.1 Grundlagen

Bei einer realen Spannungsquelle nimmt die Spannung als Funktion des Stroms ab. Um dieses Phänomen beschreiben zu können wird die Existenz eines in Serie zur Spannungsquelle geschalteten Innenwiderstand

$$R_I$$

angenommen. Im Falle eines Stromflusses fällt an diesem Innenwiderstands eine Spannung ab, die die Klemmspannung reduziert. Der Strom in der Schaltung wird nach dem Ohm'schen Gesetz berechnet:

$$I = \frac{U_0}{R_I + R_L}$$

Dieser Strom bewirkt am Innenwiderstand einen Spannungsabfall, wodurch die Klemmspannung durch

$$U_{KL} = U_0 - I * R_I$$

gegeben ist. Die Klemmspannung wird durch Belastung also linear kleiner als die Quellspannung.

```
[100]: Unsicherheit_Klemmspannung = 0.008
        Unsicherheit_Strom = 0.0015

        Klemmspannung_input = "1.566, 1.565, 1.564, 1.562, 1.556, 1.551, 1.546, 1.538, 1.533, 1.531" #in V
        Strom_input = "0.00776, 0.00964, 0.01276, 0.01895, 0.03645, 0.0473, 0.068, 0.0865, 0.097, 0.1193" #in A

        Klemmspannung = np.array([float(v.strip()) for v in Klemmspannung_input.split(",")])
        Strom = np.array([float(i.strip()) for i in Strom_input.split(",")])

        if Klemmspannung.size != Strom.size or Klemmspannung.size < 2:
            raise ValueError("Bitte gleich viele Werte eingeben und mindestens zwei Messpunkte angeben.")

        coeffs, cov = np.polyfit(Strom, Klemmspannung, 1, cov=True)
        r = coeffs[0]
        u0 = coeffs[1]

        r_std_err = np.sqrt(cov[0][0])
        u0_std_err = np.sqrt(cov[1][1])

        print(f"Steigung R = {r:.5g} A/V")
        print(f"Standardfehler _R = {r_std_err:.3g} A/V")
        print(f"U0 = {u0:.5g} V")
```

```

print(f"Standardfehler _U = {u0_std_err:.3g} V")

y_fit = r * Strom + u0

plt.figure()
plt.errorbar(Strom, Klemmspannung, xerr=Unsicherheit_Klemmspannung,
    ↪yerr=Unsicherheit_Strom, fmt='o', capsize=3, label="Fehlerbalken",
    ↪zorder=10);
plt.plot(Strom, Klemmspannung, 'o', markersize=3.5, label="Messdaten",
    ↪zorder=10)
plt.plot(Strom, y_fit, color="red", label="Linearer Fit")
plt.xlabel("Strom I in [A]")
plt.ylabel("Klemmspannung U in [V]")
plt.title("Reale Spannungsquelle")
plt.legend()
plt.grid(True)
plt.show()

```

Steigung R = -0.33439 A/V
 Standardfehler _R = 0.0126 A/V
 U0 = 1.568 V
 Standardfehler _U = 0.000799 V

Steigung R = (0,334 +/- 0,013) Ohm

U0 = (1,568 +/- 0,009) V

