Spezifische_Waermekapazitaet

July 7, 2024

1 Fakultät für Physik

1.1 Physikalisches Praktikum P2 für Studierende der Physik

Versuch P2-33, 34, 35 (Stand: April 2024)

Raum F1-08

2 Spezifische Wärmekapazität

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

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

Gruppennummer: Mo32

Betreuer: Asya Can

Versuch durchgeführt am: 17.06.24

Beanstandungen zu Protokoll Version _____:

Testiert am: _____ Testat: _____

3 Durchführung

Die Anleitung zu diesem Versuch finden Sie hier.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import kafe2
from uncertainties import ufloat, unumpy as unp
import pathlib
```

[2]:
```
# erstellen einer Funktion für kafe2 Fits
def fit_funktion(xy_data, model_function, xy_error, xy_label, title,␣
 ↪constraint=[], add_error=True):
    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)
    if add_error:
        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
```

```
def weighted_mean_gauss(arr):
    res = np.sum( unp.nominal_values(arr) / unp.std_devs(arr) ) / np.sum( 1 /␣
 ↪unp.std_devs(arr) )
    return res

def std_weighted_mean_gauss(arr):
    N = unp.nominal_values(arr).size
    arr_bar = weighted_mean_gauss(arr)
    return np.sqrt( (N/(N-1)) * np.sum( (unp.nominal_values(arr)-arr_bar)**2 /␣
 ↪unp.std_devs(arr) ) / np.sum( 1/unp.std_devs(arr) ) )

def calculate_T(V):
    return ufloat(136.1,0.54) * unp.sqrt(V + ufloat(7.587,0.035)) - ufloat(373.
 ↪9,2.3)

def correction_T(t):
    corr = ufloat(22.44,0.29) + ufloat(-207.19,0.56) * unp.exp(t /␣
 ↪ufloat(-19470,120))
    return np.abs(corr - corr[0])

# specific heat capacity of water at room temperature
c_H2O = 4184 # J/kg K
```

3.1 Aufgabe 1: Spezifische Wärmekapazität aus Mischtemperatur

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

- Bestimmen Sie so genau wie mit den verfügbaren Mitteln möglich, die spezifische Wärmekapazität c_X von Aluminium und noch eines weiteren Metalls, durch das Herstellen von Mischtemperaturen in einem Glas-Kalorimeter.
- Bearbeiten Sie hierzu die folgenden Aufgaben.

3.1.1 Aufgabe 1.1: Planung des Messvorhabens

- Beschreiben Sie das von Ihnen geplante Messvorhaben und den dazu verwendeten Aufgbau in eigenen Worten.
- Führen Sie ggf. geeignete Vergleichs- und Kalibrationsmessreihen durch, um sich von der Eignung der Methode zu überzeugen oder Korrekturen oder Unsicherheiten abzuschätzen.

The specific heat capacity (SHC) is a measure of a materials ability to store thermal energy. A SHC of $1\,\frac{J}{kgK}$ means a thermal energy of $1\,J$ is needed to heat a material of mass $1\,kg$ by $1\,K$. It is definded as follows: $c_M = \frac{1}{m_M}\frac{\partial Q}{\partial T}$.

In this experiment we want to measure the SHC of aluminium and another metal of our choice by "mixing" the metal with water inside a calorimeter. For this, we measure the masses of both the water and the metal, their starting temperatures and finally their equilibrium temperatures. There

are ceratin starting paramters that are to be fixed prior to the experiment, which will be discussed in the following.

- We used a granulate of the metal instead of a solid body so that the heat exchange can happen more efficently due to higher surface area

- For the starting temperatures we chose hot metal and water at room temperature for a few reasons: first, we need to minimize the error of heat exchange of the water with the environment, so if both have around the same temperature, heat exchange will barely happen and we keep that error low. Second, the SHC of water at room temperature is well known and stays very constant around room temperature (which is also one reason to use water and not another liquid). Due to its much higher SHC than metal, the equilibrium temperature is not much higher than its starting temperature and we can use the known, constant value. Also water is the most abundant resource on our planet and thus readily available, though we needed pure, distilled water, which is not as abundant as, say, sea water, but it's still water (that's the second reason to use water over another liquid).

- The water to metal weight ratio should definitely be in favor of the water so that, again, its temperature doesn't change too much and we can use the known, constant value for its SHC

- Since the calorimeter also has its own SHC, we will first mix room temperature water with warm water to estimate its effect on the subsequent measurements

The formula to calculate a material M's SHC is given in the preparation as the following:

$$c_M = c_{H_2O} \frac{m_{H_2O}(T_{Mix} - T_{H_2O})}{m_M(T_M - T_{Mix})}$$

To get to this result, we start with the formula form above. We can write it as $c_M = \frac{1}{m_M}\frac{\Delta Q}{\Delta T}$ because we're only interested in the absolute changes in temperature. Since we mix two materials, whatever they may be, the heat ΔQ that the hotter one gives off should ideally be completely absorbed by the other material (e.g. hot metal inside a bath of colder water), resulting in a change of temperature for both materials ΔT. Thus we can equate these ΔQ and what we receive, in the case of mixing metal with water, is the following:

$$\Delta Q_{H2O} = -\Delta Q_M$$

$$c_{H2O} \cdot m_{H2O} \cdot \Delta T_{H2O} = -c_M \cdot m_M \cdot \Delta T_M$$

$$\Leftrightarrow c_{H2O} \cdot m_{H2O} \cdot (T_{Mix} - T_{H2O}) = c_M \cdot m_M \cdot (T_M - T_{Mix})$$

where T_M and T_{H2O} are the starting temperatures and ΔT_{H2O} and ΔT_M are the absolute changes in temperature. The minus sign has already been absorbed into the left hand side of the equation. Solving for c_M yields

$$c_M = c_{H_2O} \frac{m_{H_2O}(T_{Mix} - T_{H_2O})}{m_M(T_M - T_{Mix})}$$

3.1.2 Aufgabe 1.2: Durchführung des Messprogramms und Auswertung

- Führen Sie eine Messreihe geeigneter Länge für Aluminium und ein weiteres Metall Ihrer Wahl durch.

- Bestimmen Sie für Ihre Auswertung Unsicherheiten auf alle Parameter, die die Messung Ihrer Meinung nach beeinflussen können, pflanzen Sie diese auf das Ergebnis fort und vergleichen Sie mit der Erwartung.

First, we measured the equilibrium temperature of a mix of cold and warm water, while measuring the weight and temperature of both at appropriate times to get an estimate of how the SHC of the calorimeter affects all our measurements. For that we compare the equilibrium temperatures we measured with what we would theoretically expect given the masses and starting temperatures we used. The formula can be derived from the one above for water and another, unspecified material:

$$T_{Mix} = \frac{m_{warm}T_{warm} + m_{cold}T_{cold}}{m_{warm} + m_{cold}}$$

[3]:
```
# Measurement with only water
# masses of the cold and warm water in kg
m_water_cold = unp.uarray([142.93, 130.22, 126.89, 114.38, 110.01],[.01]) *␣
 ↪10**(-3)
m_water_warm = unp.uarray([56.90, 83.13, 85.16, 76.42, 115.59],[.01]) * 10**(-3)
# mass of the calorimeter in kg
m_calorimeter = ufloat(467.98 , .01) * 10**(-3)
# temperatures of the cold and warm water and equilibrium temperatures in K
T_water_cold = unp.uarray([22.9, 23.8, 26.6, 31.2, 24.2],[.1]) + 273.15
T_water_warm = unp.uarray([59.8, 58.6, 53.5, 81.0, 63.2],[.1]) + 273.15
T_water_mix = unp.uarray([33.2, 37.1, 36.9, 49.3, 44],[.1]) + 273.15

# Expected equilibrium temperatures
def equil_temp_water(m_w, m_c, T_w, T_c):
    return ( m_w * T_w + m_c * T_c ) / ( m_w + m_c )

T_water_mix_exp = equil_temp_water(m_water_warm, m_water_cold, T_water_warm,␣
 ↪T_water_cold)
print(f"Measured equilibrium temperatures: {unp.nominal_values(T_water_mix)}")
print(f"Expected equilibrium temperatuers: {unp.
 ↪nominal_values(T_water_mix_exp)}")

# Calculate by how much every measurement deviates from the expectation
ratio_exp_to_measure = np.absolute( T_water_mix / T_water_mix_exp - 1 )
mean_ratio = weighted_mean_gauss(ratio_exp_to_measure)
print(f"Weighted average of the ratios of measured to expected equilibrium␣
 ↪temperatures: {mean_ratio} = {mean_ratio*100:.1f}%")
```

```
Measured equilibrium temperatures: [306.35 310.25 310.05 322.45 317.15]
Expected equilibrium temperatuers: [306.55698093 310.50952191 310.55313134
324.29610063 317.33231383]
Weighted average of the ratios of measured to expected equilibrium temperatures:
0.0019159309428620388 = 0.2%
```

By taking the ratio of the measured and the expected equilibrium temperatures and subtracting one, because we only want to know by how much it deviates, and taking the weighted average, we get a value of $\approx 0.002 = 0.2\%$. The measured temperature is lower than what it should be by 0.2% due to the SHC of the calorimeter and maybe other factors as well, such as minimal heat exchange with the environment.

We can calculate the SHC of the calorimeter by using the same values of the water as above. We unfortunately didn't measure the mass of the calorimeter, so we asked another group for their value that did. Its mass is $m_K = (467.98 \pm 0.01)\, kg$. We also have to slightly modify the formula for the SHC of a specific material a little to account for the SHC of the calorimeter:

$$(C_{Calori} + c_{H2O} m_{Cold})(T_{Mix} - T_{Cold}) = c_{H2O} m_{Warm}(T_{Warm} - T_{Mix})$$
$$\Leftrightarrow C_{Calori} = c_{H2O}(m_{Warm} \frac{T_{Warm} - T_{Mix}}{T_{Mix} - T_{Cold}} - m_{Cold})$$

where C_{Calori} is the heat capacity of the calorimeter.

[4]:
```
# values need to be in K (Temp), kg (mass) and J/kg*K (spec. heat cap.)
def get_cM(m_H2O, mM, T_H2O, TM, T_mix):
    return c_H2O * ( m_H2O / mM ) * ( T_mix - T_H2O ) / ( TM - T_mix )

C_calori_arr = c_H2O * ( m_water_warm * ( ( T_water_warm - T_water_mix ) / (␣
 ↪T_water_mix - T_water_cold ) ) - m_water_cold )

C_calori = ufloat( weighted_mean_gauss( C_calori_arr ) ,␣
 ↪std_weighted_mean_gauss( C_calori_arr ) )
print(f"The calorimeters heat capacity: C_Calori = {C_calori} J/K")
print(f"The calorimeters specific heat capacity: c_Calori = {C_calori/
 ↪m_calorimeter} J/kgK")
C_calori/m_calorimeter
```

```
The calorimeters heat capacity: C_Calori = 35+/-33 J/K
The calorimeters specific heat capacity: c_Calori = (8+/-7)e+01 J/kgK
```

[4]:
```
75.35004592833323+/-70.05546652303882
```

To actually get the SHC of the calorimeter, also need to divide the heat capacity by the mass which yields a SHC for the calorimeter of $c_Calori = (75 \pm 70)\, \frac{J}{kgK}$. This will be our systematic error on all following measurements of SHC in this task because it always plays a role.

Now, we first measured the SHC of aluminium. Unfortunately, since we didn't know we had to heat the aluminium (and later the tin) up to about $100°C$, we only heated it to about $60°C - 90°C$ (the tin to about $70°C - 90°C$). Because of that, we did one additional measurement with $100°C$ hot aluminium and tin respectively to check if our results are acceptable despite that error or by how much they might be off.

The formula for calculating the SHC was:

$$c_M = c_{H_2O} \frac{m_{H_2O}(T_{Mix} - T_{H_2O})}{m_M(T_M - T_{Mix})}$$

[5]:
```
# Measurements with aluminium (too cold)
# masses of the water and aluminium in kg
m_water_al = unp.uarray([182.53, 182.45, 201.10, 203.22, 204.62],[.01]) *␣
 ↪10**(-3)
m_al = ufloat(22.62,.01) * 10**(-3)
# Temperatures of the water, aluminium and equilibrium in K
T_water_al = unp.uarray([23.3, 24.1, 25.2, 25.6, 26.1],[.1]) + 273.15
T_al = unp.uarray([94.9, 86.5, 71.7, 69.2, 66.7],[.1]) + 273.15
T_mix_al = unp.uarray([25.9, 26.3, 26.6, 27.3, 27.5],[.1]) + 273.15

c_al_cold = ufloat( weighted_mean_gauss( get_cM( m_water_al , m_al , T_water_al␣
 ↪, T_al , T_mix_al ) ) , std_weighted_mean_gauss( get_cM( m_water_al , m_al ,␣
 ↪T_water_al , T_al , T_mix_al ) ) )


# Measurement with aluminium (100°C)
m_water_al_hot = ufloat(147.56, .01) * 10**(-3)
m_al_hot = ufloat(18.93, .01) * 10**(-3)
T_water_al_hot = ufloat(23.8, .1) + 273.15
T_al_hot = ufloat(105.9, .1) + 273.15
T_mix_al_hot = ufloat(27.7, .1) + 273.15

c_al_hot = get_cM( m_water_al_hot , m_al_hot , T_water_al_hot , T_al_hot ,␣
 ↪T_mix_al_hot )

print(f"Literature value of SHC of aluminium: 896 J/kg*K")
print(f"SHC of aluminium (60-90°C): c_Al = {c_al_cold} J/kg*K")
print(f"SHC of aluminium (100°C): c_Al = {c_al_hot} J/kg*K")
print(f"Ratio of aluminium (cold) to literature: {c_al_cold/896:.3f}")
print(f"Ratio of aluminium (hot) to literature: {c_al_hot/896:.3f}")
```

```
Literature value of SHC of aluminium: 896 J/kg*K
SHC of aluminium (60-90°C): c_Al = (1.29+/-0.13)e+03 J/kg*K
SHC of aluminium (100°C): c_Al = (1.63+/-0.06)e+03 J/kg*K
Ratio of aluminium (cold) to literature: 1.443+/-0.143
Ratio of aluminium (hot) to literature: 1.815+/-0.068
```

The values we get are:
- SHC of aluminium (aluminium only heated to about $60°C$–$90°C$): $c_{Al} = (1.29 \pm 0.13 \pm 0.08^{syst})\ \frac{kJ}{kgK}$
- SHC of aluminium (aluminium heated to about $100°$): $c_{Al} = (1.63 \pm 0.06 \pm 0.08^{syst})\ \frac{kJ}{kgK}$

Unfortunately, the result is bigger by about 50% than the expected literature value $c_{Al} = 0.896\ \frac{kJ}{kgK}$. One possible explanation is that we might not have allowed the metal to assume the temperature of the water it was heated in, and since we measured the temperature of the water and not the metal itself, the actual temperature might actually have been lower. The result would be a higher SHC, and the value our calculation yields supports that. By extension, the measurement at $100°C$, that should have corrected our previous measurements a little, turns out even worse than the latter, probably for the same reason, only that it would have taken even longer to assume the $100°C$

temperature. Obviously, the error resulting from heat exchange between water and calorimeter we calculated before cannot account at all for this discrepancy and even the heat capacity of the calorimeter is wildly insufficient.

[6]:
```
# Measurements with tin
# masses of the water and tin in kg
m_water_tin = unp.uarray([186.53, 206.76, 202.88, 215.14, 173.75],[.01]) *␣
 ↪10**(-3)
m_tin = ufloat(24.86,.01) * 10**(-3)
# Temperatures of the water, tin and equilibrium in K
T_water_tin = unp.uarray([23.7, 24.2, 24.5, 24.9, 25.2],[.1]) + 273.15
T_tin = unp.uarray([87.7, 83.6, 79.4, 77.6, 71.8],[.1]) + 273.15
T_mix_tin = unp.uarray([24.9, 24.9, 25.4, 25.7, 26.0],[.1]) + 273.15

c_tin_cold = ufloat( weighted_mean_gauss( get_cM( m_water_tin , m_tin ,␣
 ↪T_water_tin , T_tin , T_mix_tin ) ) , std_weighted_mean_gauss( get_cM (␣
 ↪m_water_tin , m_tin , T_water_tin , T_tin , T_mix_tin ) ) )

# Measurement with aluminium (100°C)
m_water_tin_hot = ufloat(194.12, .01) * 10**(-3)
m_tin_hot = m_tin
T_water_tin_hot = ufloat(24.2, .1) + 273.15
T_tin_hot = ufloat(99.4, .1) + 273.15
T_mix_tin_hot = ufloat(25.5, .01) + 273.15

c_tin_hot = get_cM( m_water_tin_hot , m_tin_hot , T_water_tin_hot , T_tin_hot ,␣
 ↪T_mix_tin_hot )

print(f"Literature value of SHC of tin: 221 J/kg*K")
print(f"SHC of tin (70-90°C): c_Tin = {c_tin_cold:.1f} J/kg*K")
print(f"SHC of tin (100°C): c_Tin = {c_tin_hot:.1f} J/kg*K")
print(f"Ratio of tin (cold) to literature: {c_tin_cold/221:.3f}")
print(f"Ratio of tin (hot) to literature: {c_tin_hot/221:.3f}")
```

```
Literature value of SHC of tin: 221 J/kg*K
SHC of tin (70-90°C): c_Tin = 532.1+/-74.0 J/kg*K
SHC of tin (100°C): c_Tin = 574.7+/-44.4 J/kg*K
Ratio of tin (cold) to literature: 2.408+/-0.335
Ratio of tin (hot) to literature: 2.601+/-0.201
```

The values we get are:
- SHC of tin (tin only heated to about $70°C - 90°C$): $c_{Tin} = (532.1 \pm 74.0 \pm 75^{syst}) \frac{J}{kgK}$
- SHC of tin (tin heated to about $100°$): $c_{Tin} = (574.7 \pm 44.4 \pm 75^{syst}) \frac{J}{kgK}$

This time, our result is about 2.5 times as big as the literature value $c_{Tin} = 0.221 \frac{kJ}{kgK}$. The reasons should be the same as already mentioned above. The result of the measurement at $100°C$ is again worse than the first measurements and the SHC of the calorimeter again cannot account for the discrepancy.
The explanation from before does seems sound. What is weird, though, is that both metals have a

far lower SHC than water. So by the time the water is at a temperature of, say, $100°C$, or any other temperature for that matter, the temperature of the metal should not be much different, because it takes less energy to heat the metal by the same amount it would take to heat a body of water of the same mass. Since the metal weighed much less than the water it was heated in and it had a far lower SHC, it should not have taken it long to assume the temperature of the water. The only thing that we can think of that might explain these wrong results is that water, because of its high SHC, has a tendency to store heat well. Then it would take longer than expected for the metal to heat up, despite the precautions we have taken to allow for quick and efficient heat transfer between water and metal, namely using a granulate instead of rigid body to increase surface area.

3.2 Aufgabe 2: Spezifische Wärmekapazität von Aluminium als Funktion der Temperatur

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

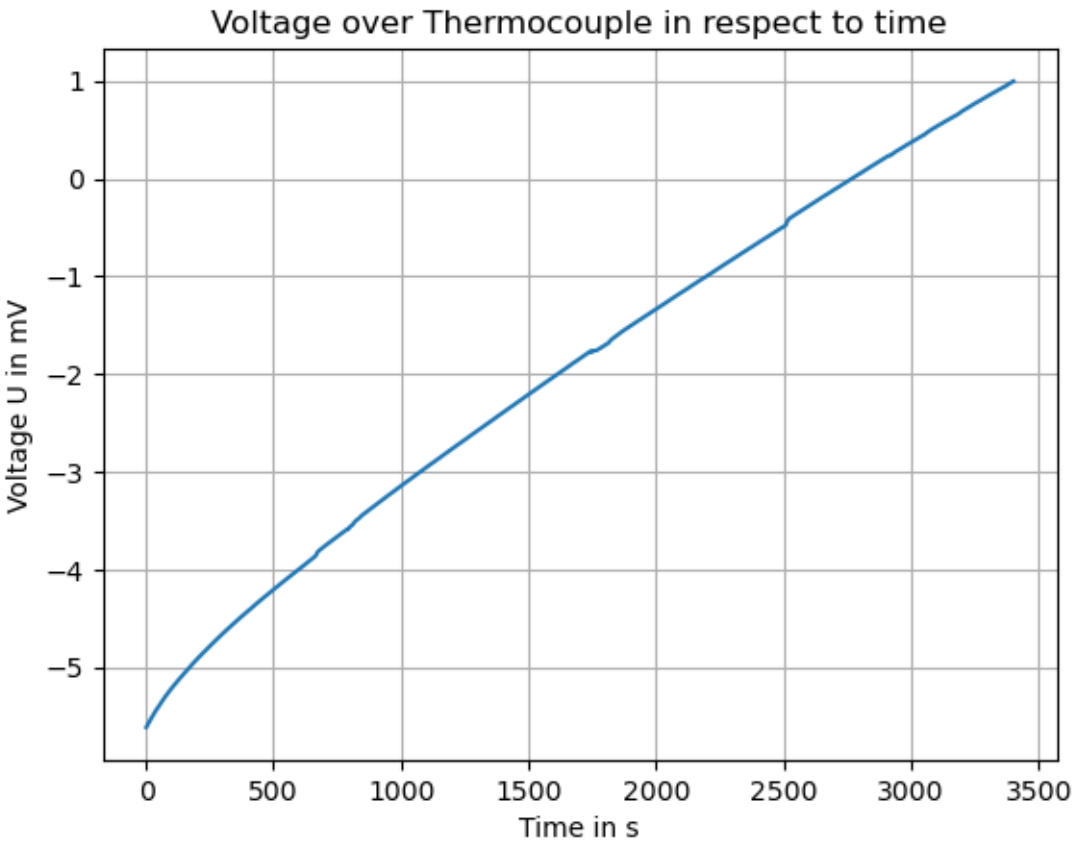
- Messen Sie $c_{\mathrm{Al}}(T)$ in Abhängigkeit von der Temperatur in einem Bereich zwischen $T = 100 - 300\,\mathrm{K}$.
- Bearbeiten Sie hierzu die folgenden Aufgaben.

3.2.1 Aufgabe 2.1: Datennahme

- Beschreiben Sie das Vorgehen in eigenen Worten.
- Bereiten Sie die Datennahme vor, nehmen Sie die Messpunkte auf.

---

[7]:
```
# reading the data
data = pd.read_csv('Messungen/Aufgabe_2_Spez_Wärme.csv')

t = np.array([float(x) for x in data['time in s']])
U = - np.array([float(x.replace(',','.')) for x in data['channel 1']])

# plotting the data
plt.plot(t,U)

plt.grid()
plt.title('Voltage over Thermocouple in respect to time')
plt.xlabel('Time in s')
plt.ylabel('Voltage U in mV')

plt.show()
```



In this task, a cylinder made from aluminium is cooled to about $-196°C$ with the help of nitrogen and then gradually brought to higher temperatures again with the help of a heatingwire that delivers a constant poweroutput. During this process, the voltage over a thermoelement is measured and can then be converted to the temperature of the aluminium. A thermoelement is a couple of two different metals joined together, which leads to a voltage changing with the heat applied from the outside. Through this data, the specific heatcapacity of aluminium can be calculated.

In the graph above, the unprocessed data is shown.

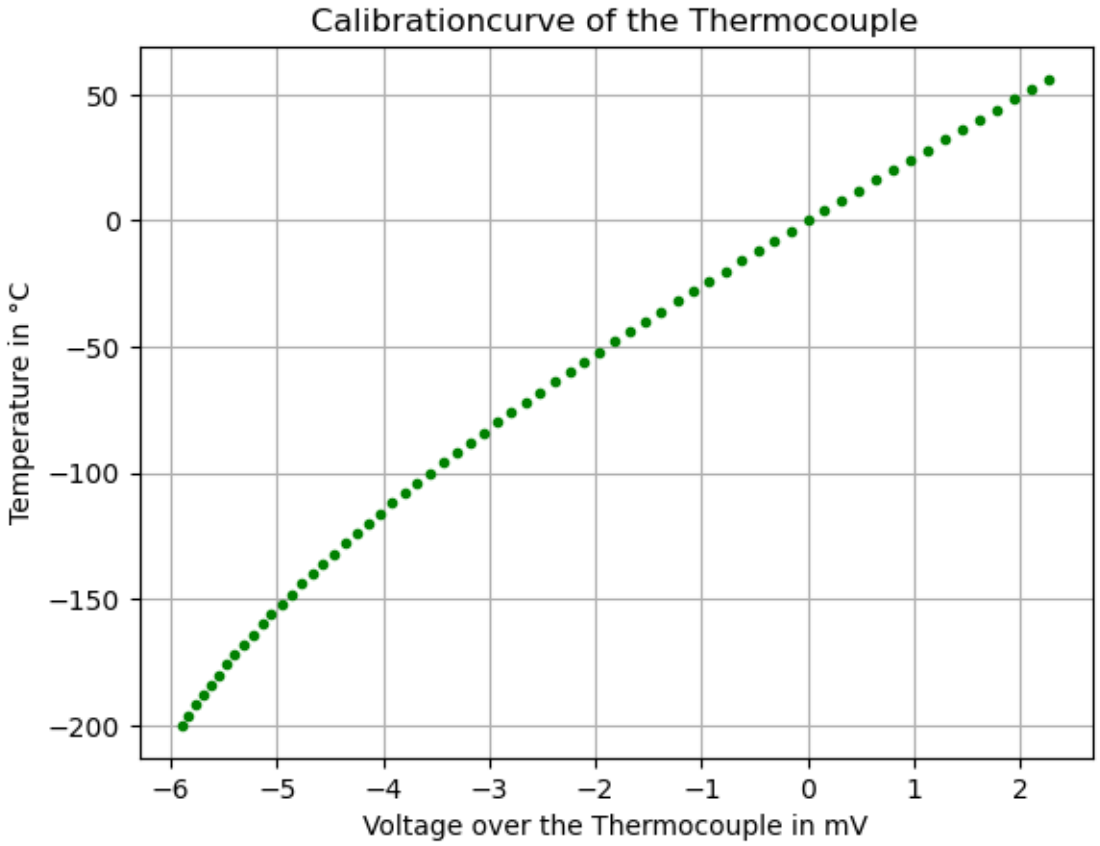
3.2.2 Aufgabe 2.2: Kalibration des Thermoelements

- Kalibrieren Sie das NiCr-Ni-Thermoelement von ursprünglichen Angabe von Volt auf eine Temperaturmessung in Kelvin.
- Sie können diese Kalibration mit Hilfe der Datei calibration.csv vornehmen.

---

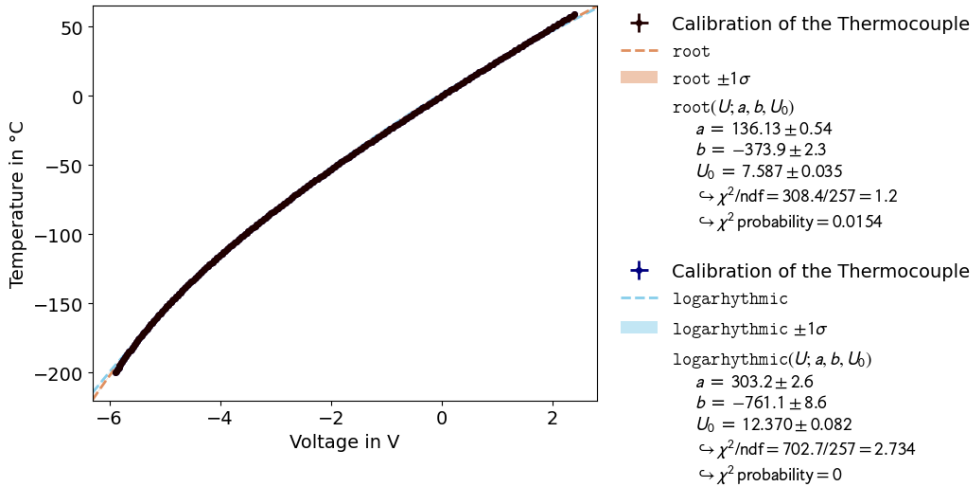
[8]:
```
# reding the data
data = pd.read_csv('params/calibration.csv')
```

```
# plottting the raw data
T_calib = data['T']
U_calib = data['U']

plt.plot(U_calib[::4],T_calib[::4],'g.')

plt.title('Calibrationcurve of the Thermocouple')
plt.xlabel('Voltage over the Thermocouple in mV')
plt.ylabel('Temperature in °C')
plt.grid()

plt.show()
```



[9]:
```
# defining different modelfunctions
def logarhythmic(U,a=300,b=-750,U_0=12):
    return a*np.log(U+U_0) + b

def root(U,a=136,b=-370,U_0=7.5):
    return a*np.sqrt(U+U_0) + b
```

```
# defining the errors
T_calib_err = 1
U_calib_err = 0.01

# doing three different fits
xy_data = kafe2.XYContainer(U_calib,T_calib)
xy_data.label = 'Calibration of the Thermocouple'
xy_data.axis_labels =('Voltage in V','Temperature in °C')

root_fit = kafe2.XYFit(xy_data,root)
root_fit.add_error('x',U_calib_err)
root_fit.add_error('y',T_calib_err)
root_fit.do_fit()

logarhythmic_fit = kafe2.XYFit(xy_data,logarhythmic)
logarhythmic_fit.add_error('x',U_calib_err)
logarhythmic_fit.add_error('y',T_calib_err)
logarhythmic_fit.do_fit()

# plotting the fits results
plot = kafe2.plot([root_fit, logarhythmic_fit])

plt.show()
```



In this task, the function to convert the voltage over the thermocouple into temperature will be defined with the help of previously measured data.

Whilst firstly looking at the data in the Graph shown above, it was found, that maybe a rootfunction or a logarhythm would match best in a fit. Thus, the functions $T = a \cdot \sqrt{U + U_0} + b$ and $T = a \cdot ln(U + U_0) + b$ will be used. Here, the uncertainties chosen are $\sigma_T = 0.1$ and $\sigma_U = 0.01$. Because

$\chi^2 = 2.734$ for the logarhythm and $\chi^2 = 1.2$ for the root, the root is chosen and in the following, the function used to convert voltage to temperature is:

$$T = (136.13 \pm 0.54)\sqrt{U + (7.587 \pm 0.035)}°C - (373.9 \pm 2.3)°C$$

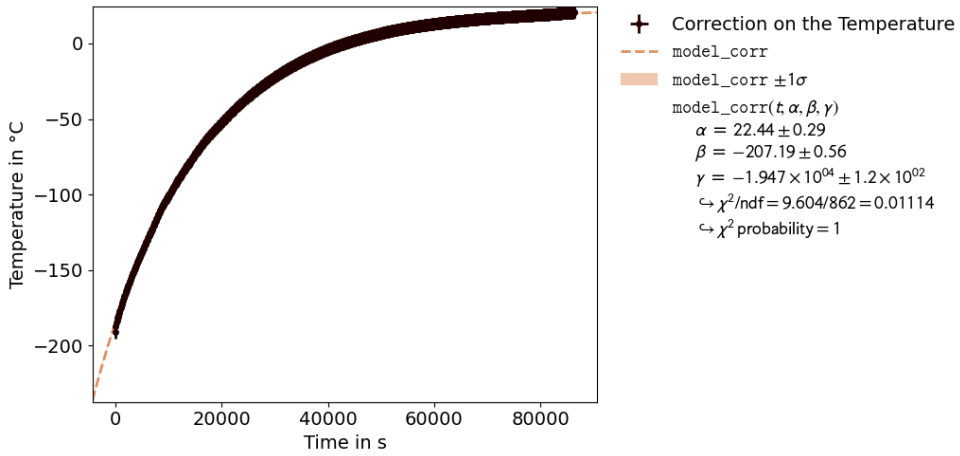
3.2.3 Aufgabe 2.3: Korrektur des Wärmegangs

Trotz Wärmeisolation nimmt der für die Messung verwendete Aluminium-Hohlzylinder im Verlauf der Messung zusätzlich zur elektrischen Heizleistung Wärme aus der Umgebung auf (**Wärmegang**).

- Sie könnten den Wärmegang durch eine zweite, gleichartige Messung, ohne elektrische Heizung abschätzen (**Nullmessung**) und korrigieren. Eine solche Messung dauert allerdings 24 Stunden, sie lässt sich also nicht an einem Versuchstag durchführen.
- Sie können stattdessen die Daten aus der Datei waermegang.csv für diese Korrektur verwenden.

---

[10]:
```
# reading the data
data = pd.read_csv('params/waermegang.csv')

T_diss = [x.nominal_value for x in calculate_T(data['U'])][::10]
t_diss = data['t'][::10]

T_diss_err = np.array([x.std_dev for x in calculate_T(data['U'])])[::10] + 1

# definign the model_function
def model_corr(t,alpha=20,beta=-200,gamma=-20000):
    return alpha + beta * np.exp(t/gamma)

# making a fit
xy_data = kafe2.XYContainer(t_diss,T_diss)
xy_data.label = 'Correction on the Temperature'
xy_data.axis_labels = ('Time in s','Temperature in °C')

corr = kafe2.XYFit(xy_data,model_corr)
corr.add_error('y',T_diss_err)
corr.do_fit()

plot = kafe2.plot(corr)
plt.show()
```



Here, we correct the Temperature in our data for the heat, radiated to the sorroundings. This is done, by cooling the cylinder, letting it warm up by itself and measuring the voltage over the Thermocouple. Because the mentioned process takes about $24h$, the data used was obtained in advance.

We now fit the model $T_0 = \alpha + \beta e^{\frac{t}{\gamma}}$ to obtain a function, that can later be subtracted from the measured temperature whilst heating the cylinder. The result we get is:

$$T_0 = (22.44 \pm 0.29)°C + (-207.19 \pm 0.56) \cdot exp(\frac{t}{(-19470 \pm 120)})°C$$

The Fit itself is in an acceptable range, because $\chi^2 = 0.0202$ is very low and the model can't be further simplified.

3.2.4 Aufgabe 2.4: Bestimmung von $c_{\mathrm{Al}}(T)$

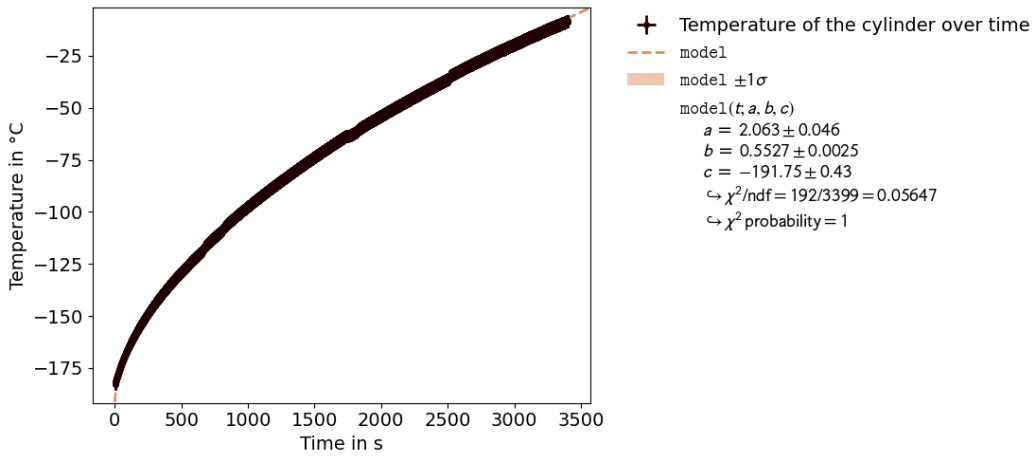
- Bestimmen Sie den Verlauf der spezifischen Wärmekapazität von Aluminium als Funktion von T.
- Bestimmen Sie aus diesem Verlauf den Wert für $T = 20°\mathrm{C}$ und vergleichen Sie mit Ihrer Erwartung.
- Bestimmen Sie aus diesem Verlauf den Wert für T_{Mix} und vergleichen Sie mit dem Ergebnis aus **Aufgabe 1.2**.

---

[11]:
```
# calculating and correcting the T-array
T = np.array([x.n for x in calculate_T(U)-correction_T(t)])
T_err = np.array([x.std_dev for x in (calculate_T(U) - correction_T(t))])

# defining a modelfunction
def model(t,a=1.5,b=0.6,c=-200):
    return a*t**b + c
```

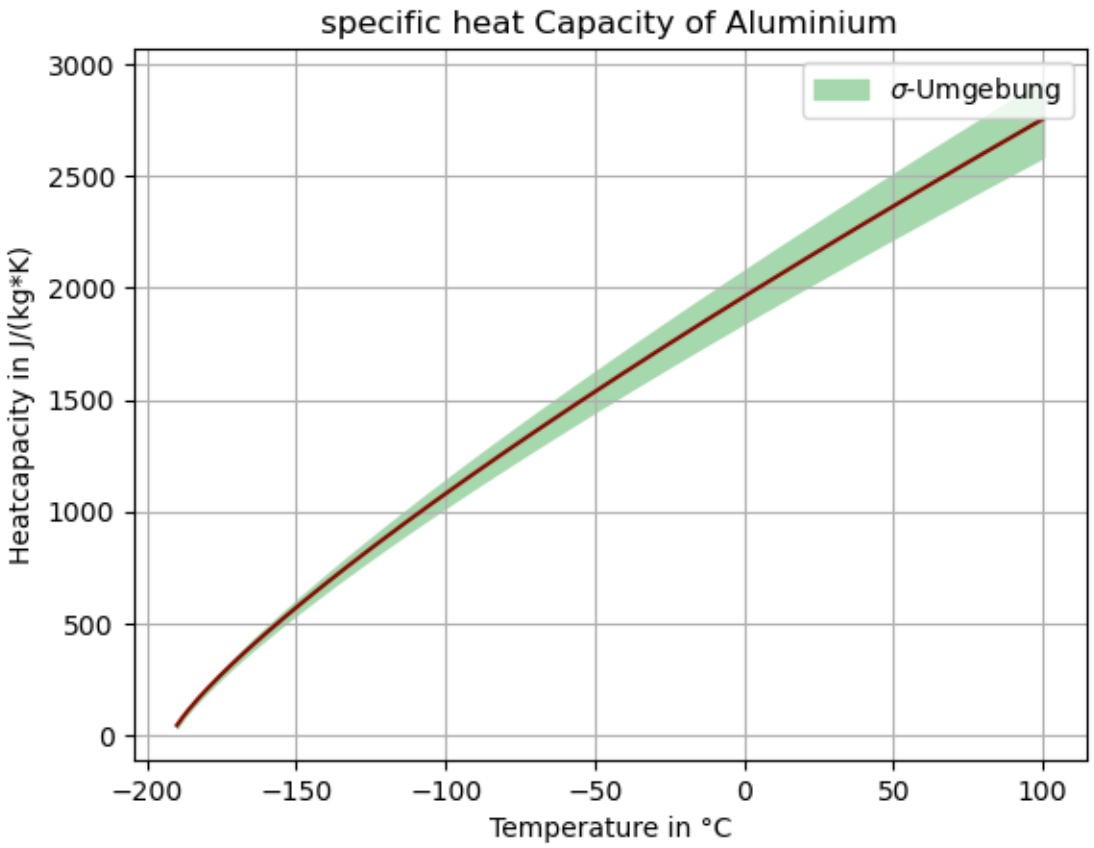
```
# Fitting a temperature over time curve
xy_data = kafe2.XYContainer(t,T)
xy_data.label = 'Temperature of the cylinder over time'
xy_data.axis_labels = ('Time in s','Temperature in °C')

fit = kafe2.XYFit(xy_data, model)
fit.add_error('y',T_err)
fit.do_fit()

plot = kafe2.plot(fit)
plt.show()

# putting the parameters and the errors in arrays
params = fit.parameter_values
errs = fit.parameter_errors

# defining a function to get T' over T
def get_Tbar(T,a,b,c):
    return a*b* ((T-c)/a) ** ((b-1)/b)

# defining a function to get the heatcapacity
def get_cm(T,a,b,c):
    m = 0.338
    P = 19.3
    return P / (m * (get_Tbar(T,a,b,c)))
```

```
C:\Users\probz\AppData\Local\Temp\ipykernel_11528\1386504532.py:7:
RuntimeWarning: invalid value encountered in power
  return a*t**b + c
```



[12]:
```
# defining the temperature T and the number of valuess for monte-carlo
T_lin = np.linspace(-200,100,300)
N = 10000

# making the monte-carlo simulation
rng = np.random.default_rng(42)
a_rnd,b_rnd,c_rnd = rng.normal(params, errs, size = (N,3)).transpose()
T_mul = np.tile(T_lin,(N,1)).transpose()

# calculating the result of monte-carlo
res = get_cm(T_mul,a_rnd,b_rnd,c_rnd)

res_mean = np.mean(res,axis=1)
res_std = np.std(res,axis=1)

# plotting the results
plt.grid()
plt.plot(T_lin,res_mean,color='#820900')
plt.
  ↪fill_between(T_lin,(res_mean+res_std),(res_mean-res_std),color='#A6D8AD',label='$\sigma$-Um
plt.legend()
plt.title('specific heat Capacity of Aluminium')
plt.xlabel('Temperature in °C')
plt.ylabel('Heatcapacity in J/(kg*K)')
plt.show()

# printing at 20°C
print(f'the specific heatcapacity of Aluminium at 20°C is c(Aluminium) =␣
  ↪{res_mean[219]}+/-{res_std[219]}')
print(f'the factor between task 1.2 and 24 is c(1.2)/c(2.4) = {c_al_cold/
  ↪ufloat(res_mean[219],res_std[219])}')
print(f'the factor between task 2.4 and literature is c(lit)/c(2.4) =␣
  ↪{ufloat(res_mean[219],res_std[219])/896}')
```

```
C:\Users\probz\AppData\Local\Temp\ipykernel_11528\1386504532.py:27:
RuntimeWarning: invalid value encountered in power
  return a*b* ((T-c)/a) ** ((b-1)/b)
```




```
the specific heatcapacity of Aluminium at 20°C is c(Aluminium) =
2124.204663883034+/-126.11692752324718
the factor between task 1.2 and 24 is c(1.2)/c(2.4) = 0.61+/-0.07
the factor between task 2.4 and literature is c(lit)/c(2.4) = 2.37+/-0.14
```

The task is finalized by calculating the specific heatcapacity of aluminium. For that, we take the measured voltage and convert it to the temperature by using the funciotn mentioned in task 2.2. Forthermore the Temperature is corrected for the heat radiated to the surroundings according to the formula obtained in task 2.3. To this Temperature, the model $T(t) = a \cdot t^b + c$ is fitted. By inverting this model, it follows that $t(T) = (\frac{T-c}{a})^{\frac{1}{b}}$ an through deriving the term for $T(t)$ and substituting the final result is:

$$\dot{T}(T) = ab \cdot (\frac{T-c}{a})^{\frac{b-1}{b}}$$

$$c_{Al} = \frac{P_H}{m_{Al} \cdot \dot{T}(T)}$$

The fit in which the parameters are determined is acceptable with a very low χ^2. The errors on a,b and c are carried on by a Monte-Carlo-Method. In this method, there are $N = 10000$ random numbers associated to the parameters following a gaussian. The mean and the standart deviation of the distribution are the values determined in the fit. With these random values the heatcapacity

is calculated and the mean and the standart deviation follow. This is then graphically shown in the figure above.

For $20°C$ we get $c_{Al} = (2.124 \pm 0.126) \frac{kJ}{kg \cdot K}$ which deviates from task 1.2 by a factor of 0.61 ± 0.07. The value calculated here is much more apart from literature, deviating by a factor of 2.37 ± 0.14.
