In [1]:
```python
import math

from metas_unclib import *

from Metas.UncLib.LinProp import UncBudget

import pandas as pd

import itables

itables.init_notebook_mode(all_interactive=True)
```

In [2]:
```python
import metas_unclib as mu
```

In [3]:
```python
def unc_budget(unc_item, show_table=True):
    tree = UncBudget.ComputeTreeUncBudget(unc_item.net_object)

    table = pd.DataFrame(
        columns=("description", "uncertainty component", "uncertain
        index=range(len(tree)+1)
    )

    for i, elem in enumerate(tree):
        table.loc[i] = (
            elem.get_Description(),
            elem.get_UncComponent(),
            elem.get_UncPercentage(),
        )
    table.loc[len(tree)] = (
        "SUMMARY",
        unc_item.stdunc,
        100.,
    )

    return table.sort_values("uncertainty percentage", ascending=Fa
```

In [4]:
```python
def tolerance(value, a):
    """
    producer tolerance of value +/- a

    returns UniformDistribution(value - a, value + a)
    """
    return UniformDistribution(value - a, value + a)
```

C_{Dewar}

$$C_{Dewar} = C_{Sys,kal} - C_W = \frac{UI}{b} - m_W c_W^{sp}$$

c_L^{sp}: spezifische Wärmekapaziät

C: Wärmekapazität Wasser

m: Masse Lösungsmittel

U, I: Spannung, Strom Heizung

b: Steigung aus T-t-Diagramm

In [5]:
```
ref_V = ufloatfromdistribution(
  tolerance(0.1, 0.0001),  # L
  desc='reference volume / L'
)
ref_V
```

Out[5]: 0.1 ± 5.773502691896423e-05

In [6]:
```
# rho water
water_rho = ufloat(0.9982, desc="water rho / kg/L")
water_m = water_rho * ref_V
water_m
```

Out[6]: 0.09982 ± 5.76311038705101e-05

In [7]:
```
water_c_sp = ufloat(4182, desc="water specific heat capacity / J/K/
```

In [8]:
```
U = ufloatfromdistribution(
  tolerance(10.0, 0.05),  # V
  desc='voltage / V'
)
I = ufloatfromdistribution(
  tolerance(1.61, 0.005),  # V
  desc='current / A'
)
P = U * I
P
```

Out[8]: 16.1 ± 0.0547121254080546

In [72]:
```
# Janosch

water_slope_values = np.array((0.027875, 0.02771894, 0.02757587))
water_slope = ufloatfromsamples(
    water_slope_values,
    desc='slope of water calibration / K/s'
)
water_slope
```

Out[72]: 0.027723269999999998 ± 0.00018962430800640334

```
# check for Samuel

water_slope_values = np.array((0.02343122, 0.02377467,
0.02358517))
water_slope = ufloatfromsamples(
    water_slope_values,
    desc='slope of water calibration / K/s'
)
water_slope
```

In [73]:
```
dewar_C_array = U.value * I.value / water_slope_values - water_m.va
dewar_C_array
```

Out[73]:
```
array([160.13123534, 163.38305149, 166.39653356])
```

In [78]:
```
dewar_C = U * I / water_slope - water_m * water_c_sp
#dewar_C.mean()
dewar_C
```

Out[78]:
```
163.2923336505831 ± 4.441980282611897
```

In [77]:
```
ufloatfromsamples(dewar_C_array)
```

Out[77]:
```
163.30360679551532 ± 3.971403791973842
```

c_L^{sp}

$$c_L^{sp} = \frac{C_L}{m} = \frac{C_{Sys,L} - C_{Dewar}}{m} = \frac{UI}{bm} - \frac{C_{Dewar}}{m}$$

```
# even more brute force for other estimates
# dewar_C = ufloat(dewar_C_array.mean(), 1.4, desc="dewar_C")
```

In [68]:
```
# Janosch
sol_slope = ufloat(
    0.0398696981455289,
    2.96817283935158E-05,
    desc="slope of solution / K/s"
)
sol_slope
```

Out[68]:
```
0.0398696981455289 ± 2.96817283935158e-05
```

```
# check for Samuel
sol_slope = ufloat(
    0.025,  # estimated (fitted to reach target result)
    2.96817283935158E-05,
    desc="slope of solution / K/s"
)
sol_slope
```

In [79]:
```
# since the scale is very accurate in comparison to the other uncer
sol_m_perV = ufloat(
  0.08236242,  # kg
  desc='solution mass / kg'
)
```

In [80]:
```
# absolute heat capacity of ethanol
U * I / sol_slope - dewar_C
```

Out[80]:
```
240.5231138913868 ± 4.035878115463858
```

In [81]:
```
# specific heat capacity
sol_c_sp = (U * I / sol_slope - dewar_C) / sol_m_perV
sol_c_sp
```

Out[81]:
```
2920.301684814346 ± 49.00145133501247
```

In [66]:
```
unc_budget(sol_c_sp)
```

Out[66]:

	description	uncertainty component	uncertainty percentage
5	SUMMARY	77.177717	100.0
3	slope of water calibration / K/s	76.545151	98.367472
2	slope of solution / K/s	9.283376	1.446865
1	reference volume / L	2.926253	0.143761
4	voltage / V	1.342025	0.030237
0	current / A	0.833556	0.011665

In [23]:
```
print(unc_budget(sol_c_sp).to_latex())
```

```
\begin{tabular}{llll}
\toprule
{} &                       description & uncertainty component & u
ncertainty percentage \\
\midrule
5 &                           SUMMARY &             49.001451 &
100.0 \\
3 &  slope of water calibration / K/s &             48.228291 &
96.869234 \\
4 &                       voltage / V &               6.20108 &
1.601463 \\
0 &                       current / A &              3.851603 &
0.617824 \\
2 &           slope of solution / K/s &              3.650061 &
0.554859 \\
1 &              reference volume / L &              2.926253 &
0.35662 \\
\bottomrule
\end{tabular}
```