Physikalisch-chemisches Praktikum
ETH Zürich, HS23

Measuring conductivity of water, sodium carbonate and potassium carbonate solutions under varying conditions

Janosch Jörg
DCHAB
jjoerg@ethz.ch

Maria Zimmermann
DCHAB
mazimmerm@ethz.ch

Assistant:
Toutoudaki Eirini
eirini.toutoudaki@phys.chem.ethz.ch

Abstract

The specific conductivity κ of several water samples was measured and compared. Next, the temperature dependence of a 0.1 M potassium carbonate solution was examined, yielding a thermal coefficient $\alpha_{K_2CO_3} = 0.021\,43 \pm 0.000\,08\,K^{-1}$. The specific conductivity was measured again for 0.01 M solutions of potassium carbonate and sodium carbonate, and the obtained results were subsequently used to calculate the respective molar conductivity, receiving results of $\Lambda_{K_2CO_3} = 216 \pm 4\,S\,cm^2/mol$ and $\Lambda_{Na_2CO_3} = 131.1 \pm 0.6\,S\,cm^2/mol$ respectively. By measuring κ of sodium carbonate solutions of varying concentrations, the limiting molar conductivity could be extrapolated at $\Lambda^0_{Na_2CO_3} = 251 \pm 7\,S\,cm^2/mol$. Via conductometric titration, the unknown concentration of a potassium carbonate solution was found to be $9.2 \pm 0.3\,mM$.

Zurich, January 26, 2024

Janosch Jörg Maria Zimmermann

1 Introduction

The specific conductivity κ of a solution describes the availability and capacity of movable charges within a substance. In the case of solutions, which were the focus of the experiments carried out, this refers to the ions of dissociated electrolytes.

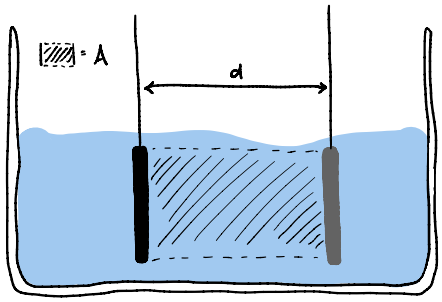

Figure 1: A schematic setup to measure the conductivity of a liquid.

Specific conductivity can be quantified by

$$\kappa = \frac{d/A}{R} \tag{1}$$

where d is the distance between between two electrodes immersed in the solution, A is the spanned cross sectional area. see fig and R the electrical resistance. It is measured in Siemens S which is equivalent to $1/\Omega$.

As such, κ correlates with how fast the available ions can move within the solution by:

$$v_i = \frac{eE}{6\pi\eta}\frac{z_i}{r_i} \tag{2}$$

whereby $z_i * e$ is the respective ion's charge, E is the electric field strength, η is the viscosity coefficient and r_i is the radius of the ion plus the solvent molecules surrounding it. It can be seen, that v_i increases when r_i or η decreases.

This explains the correlation between temperature and conductivity, as the viscosity η decreases when the temperature increases, causing v_i and therefore κ to increase as well.

Past empirical observations have shown, that within the range of several tens of degrees Celsius relative to a starting temperature T_0, changes in conductivity follow a linear model:

$$\kappa(T) = \kappa(T_0)(1 + \alpha(T - T_0)) \tag{3}$$

where α is called temperature coefficient and can be calculated by:

$$\alpha = \frac{\mathrm{d}\kappa/\mathrm{d}T}{\kappa(T_0)} \tag{4}$$

Furthermore, κ also depends on the concentration of the given electrolyte within the solution. Intuitively, the more mobile charges are available, the higher the solution's conductivity will be. This holds especially true for diluted solutions of strong electrolytes (at constant temperature) where the correlation between conductivity κ and concentration c can be described approximately by

$$\kappa = \Lambda c \tag{5}$$

where Λ is called the molar conductivity and used to compare the conductivities of different electrolytes regardless of c.

However, when measuring molar conductivities accurately over a broad range of concentrations, it becomes clear that the molar conductivity varies slightly.

Here *Kohlrausch's square root law* offers a more accurate mathematical model:

$$\Lambda = \Lambda_0 - k\sqrt{c} \tag{6}$$

where k is a constant dependant on the valency of the ions and Λ^0 is the extrapolated value generated by approximating $c = 0$ called the limiting molar conductivity.

The overall conductivity of a solution containing more than one electrolyte depends on the respective electrolyte's conductivity as well as the ratio at which they are present within the solution. As such, changes to this ratio can result in changes to the overall conductivity as is utilized in conductometric titrations.

2 Experimental

2.1 Chemicals

The following experiments were conducted using sodium carbonate, anhydrous, (105.99 g/mol), from `Sigma Aldrich`, puriss., (99.5 %, calc. to the dried substance), and potassium carbonate (138.2 g/mol), from `Sigma Aldrich`, (99 %). All chemicals were used forgoing any further purification.

Additionally, a premixed $\approx$ 0.1 M calcium chloride solution was used. Its origin and purity are unknown to the authors.

2.2 Procedure

A total of 5 experiments were prepared and carried out as follows:

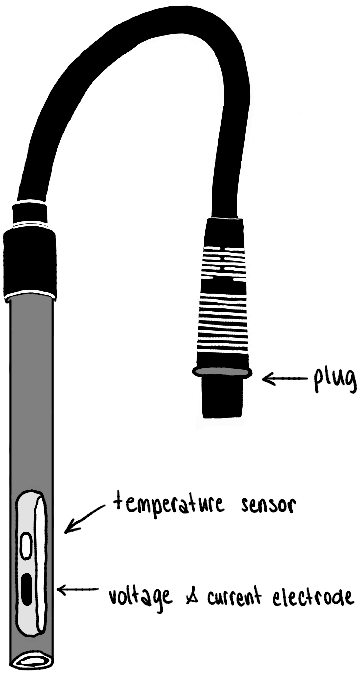

Figure 2: The `KNICK SE 204` measuring cell can measure values ranging from 1.00 µS/cm - 500 mS/cm and has a cell constant of approximately 0.5 cm^{-1}

2.2.1 Specific conductivity of water

First, the specific conductivity κ of differently treated water samples was measured using a `KNICK 703` conductometer with a `KNICK SE 204` 4-pole measuring cell (Fig. 2). Three samples were prepared for deionised water, ultrapure deionised and degassed water (from a `HUBER and CO. AG` TKA-GenPure), and tap water each and the respective conductivity measured immediately after preparation (especially for deionised and degassed water) to ensure that the results would not be falsified by any compounds formed through contact with the surrounding air.

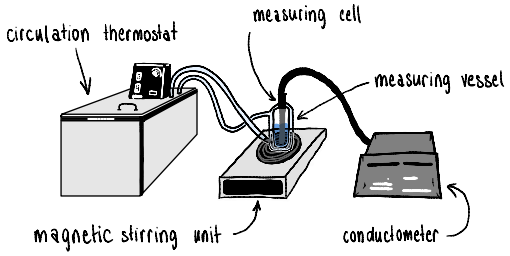

Figure 3: The entire setup for conductivity measurements, including the `LAUDA MT` circulation thermostat, the `KNICK SE 204` measuring cell and the `KNICK Conductometer 703`.

2.2.2 Correlation between temperature and conductivity

For the next experiment, the specific conductivity κ of a 0.1 M potassium carbonate solution was measured at steadily decreasing temperature. The solution was prepared by weighing 1.3895 g of sodium carbonate in a 100 mL volumetric flask using a `METTLER TOLEDO AG204 Delta Range` analytical balance and filling the flask to its mark with deionized water. The sample was transferred to the measuring vessel (Fig. 3) and heated to 50 °C by means of a `LAUDA MT` circula-

tion thermostat. The specific conductivity at 50 °C was noted in the lab journal and subsequently the heat supply to the circulation thermostat cut off, simultaneously activating the water cooler to bring the temperature of the sample to a gradual, steady decline. Finally, κ of potassium carbonate in a 0.1 M aqueous solution was noted down for temperatures between 50 °C and 25 °C in steps of 1 °C.

2.2.3 Molar conductivity of different electrolytes

Next, the molar conductivity Λ of 0.01 M solutions of potassium carbonate and sodium carbonate was calculated using values obtained from measurements of κ at constant temperature for both solutions. The potassium carbonate solution was prepared by taking 50 mL of the solution used in the previous experiment and diluting it with water at a 1:10 ratio. The sodium carbonate solution was prepared similarly to the potassium carbonate solution in the previous experiment, by weighing 0.530 g of sodium carbonate in a 100 mL volumetric flask and filling it to its mark. Samples of both solutions were transferred to the measuring vessel separately and their respective conductivity noted down after brief stirring. This was repeated twice more for each solution.

2.2.4 Concentration dependency of conductivity and limiting molar conductivity

For the next experiment, the specific conductivity κ of a 0.01 M sodium carbonate solution was measured at constant temperature and increasing concentration c. First, a 10 mL burette was rinsed out and then filled with the same solution as in the previous experiment. Then the measuring vessel was filled with 100 mL of deionized water (measured by means of a volumetric flask) and its intrinsic conductivity documented. Finally, the sodium carbonate solution was added in steps of 0.5 mL and the conductivity of the resulting solution noted down after each increase until a total of 10 mL had been added.

2.2.5 Conductometric titration

For the last experiment, a potassium carbonate solution of unknown concentration was provided by the laboratory assistant, of which exactly 100 mL were transferred to the measuring vessel via volumetric flask. A `METROHM 775` dosimat was filled with the titer, for which a pre-prepared 0.099 97 M calcium chloride solution was provided. The titer was released into the potassium carbonate solution via "Go" button in approximately 0.5 mL steps, noting down the exact volume that was added and the corresponding concentration after every increment. A total of 20 m calcium chloride was added to the solution.

3 Results and Discussion

The analysis and calculations were conducted using the R programming language [3] and Python Jupyter Notebooks [2]. The scripts are included in the appendix of this report. A.1 All uncertainties stated are provided for a 95 % confidence interval. The uncertainties were modelled with the aid of `METAS UncLib`, a Python uncertainty modelling library by the Swiss institute of metrology METAS.[5]

3.1 Conductivity of differently treated water samples

For the different types of water specific conductivities were found to be:

water treatment	κ / µS/cm
tap	283(14)
deionized	2.85(12)
degased	1.1(3)

Table 1: Specific conductivities κ of differently treated water samples.

The measured specific conductivity for

the purified, degassed water exceeds the manufacturer specifications of 0.055 µS/cm by a substantial amount.[4] To accurately measure such low conductivities, a closed measurement apparatus, where no oxygen is absorbed, would be required.

3.2 Correlation between temperature and conductivity

The thermal coefficient of the potassium carbonate conductivity was found to be $\alpha_{K_2CO_3} = 0.021\,43 \pm 0.000\,08\,K^{-1}$, and thus closely matching the approximately $0.02\,K^{-1}$ typical for aqueous solutions.[1]

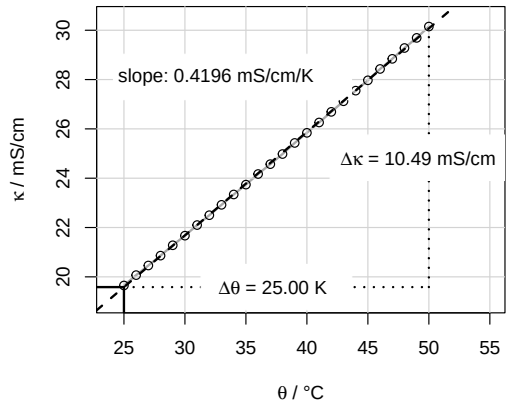

Figure 4: The specific conductivity of potassium carbonate as a function of the temperature of the solution. The correlation is perfectly linear ($R^2 > 0.9999$).

3.3 Molar conductivity of different electrolytes

From three conductivity measurements of 0.01 M K_2CO_3 and Na_2CO_3 each, molar conductivities of $\Lambda_{K_2CO_3} = 216 \pm 4\,S\,cm^2/mol$ and $\Lambda_{Na_2CO_3} = 131.1 \pm 0.6\,S\,cm^2/mol$ respectively were determined.

By recording conductivities at varying concentrations of Na_2CO_3, the molar conductivity extrapolation to $c = 0\,M$ returns $\Lambda^0_{Na_2CO_3} = 251 \pm 7\,S\,cm^2/mol$.

Using the same regression model to predict a molar conductivity for 0.01 M Na_2CO_3 yields $\Lambda_{Na_2CO_3} = 142 \pm 9\,S\,cm^2/mol$, thus confirming the above recordings with just a slight variation, which is probably due to the differences in the experimental setup.

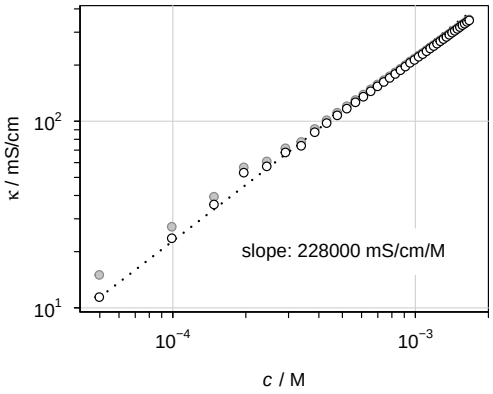

Figure 5: Plotting the specific conductivity κ against the concentration c confirms the proportional relationship (dotted line), and thus legitimizes the consistency of the molar conductivity across the entire range of dilution.
The grey points show the raw conductivity recordings, the white points are corrected for the conductivity of the pure water. For further processing, the corrected values were used.

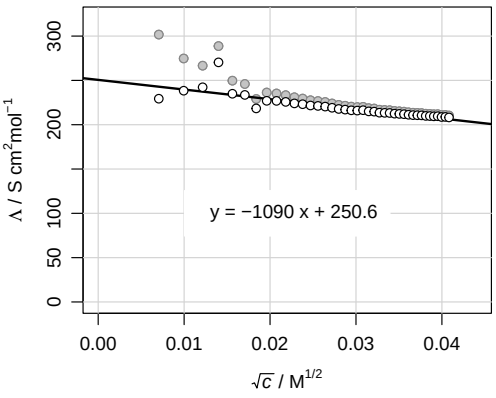

Figure 6: Extrapolation to the limiting molar conductivity in accordance with *Kohlrausch's square root law.* (Eq. 6)

3.4 Conductometric titration

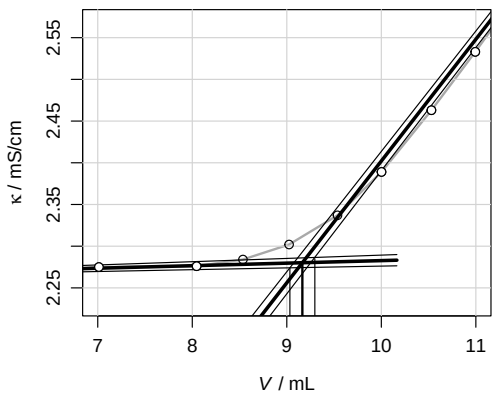

Figure 7: Magnified view of the equivalence point in the titration of K_2CO_3 with 0.099 97 M $CaCl_2$. The transparent points were not included in the calculation of the intersection point. The full titration plot can be seen in the appendix.8

Intersecting the lines before and after the equivalence points of the conductometric titration yields a titer volume $V_t = 9.2 \pm 0.3\,\text{mL}$, which corresponds to an initial concentration of $9.2 \pm 0.3\,\text{mM}$ of the K_2CO_3 solution.

4 Coursebook exercises

4.1 Ion velocity

Using equation 2:

$$v_i = \frac{eE}{6\pi\eta}\frac{z_i}{r_i} \tag{7}$$

yields $v_{Mg^{2+}} = 1.70\,\mu\text{m/s}$ and $v_{CH_3COO^-} = 0.85\,\mu\text{m/s}$.

Mg^{2+} moves faster than CH_3COO^- due to its higher charge (and smaller size, which is not even taken into account in this exercise). A speed of micrometers per second seems small at first glance, but is really fast compared to the ions size and the density of particles in a liquid.

4.2 Molar conductivity of pure water

$$\Lambda \approx \Lambda^0 = \lambda_+^0 + \lambda_-^0 = 547\,\frac{\text{S}\,\text{cm}^2}{\text{mol}}$$

$$pH = 7 \implies [H^+] = 1 \cdot 10^{-7}\,\frac{\text{mol}}{\text{L}}$$

$$= 1 \cdot 10^{-10}\,\frac{\text{mol}}{\text{cm}^3}$$

$$\kappa = \Lambda c = 547\,\frac{\text{S}\,\text{cm}^2}{\text{mol}} 1 \cdot 10^{-10}\,\frac{\text{mol}}{\text{cm}^3}$$

$$= 5.47 \cdot 10^{-8}\,\frac{\text{S}}{\text{cm}}$$

4.3 Spagetthi water

$$8\,\text{g NaCl in water} \mathrel{\widehat{=}} 0.137\,\frac{\text{mol}}{\text{L}} = 137\,\frac{\text{mol}}{\text{cm}^3}$$

$$\Lambda \approx \Lambda^0 = \lambda_+^0 + \lambda_-^0 = 127\,\frac{\text{S}\,\text{cm}^2}{\text{mol}}$$

$$\kappa = \Lambda c = 127\,\frac{\text{S}\,\text{cm}^2}{\text{mol}} 137\,\frac{\text{mol}}{\text{cm}^3}$$

$$= 0.928\,\frac{\text{S}}{\text{cm}}$$

4.4 Limiting molar conductivity HCl and KOH

$$\Lambda^0_{\text{HCl}} = \lambda^0_{\text{H}^+} + \lambda^0_{\text{Cl}^-} = 426\,\frac{\text{S}\,\text{cm}^2}{\text{mol}}$$

$$\Lambda^0_{\text{KOH}} = \lambda^0_{\text{K}^+} + \lambda^0_{\text{OH}^-} = 271\,\frac{\text{S}\,\text{cm}^2}{\text{mol}}$$

The calculated values are slightly higher than those extrapolated in Fig. 7.4 of the coursebook.

4.5 Reading the titration curve

$V_t = 18.30 \pm 0.25\,\text{mL}$
$\mathrel{\widehat{=}} 1.800 \pm 0.025\,\text{mmol}$
$\mathrel{\widehat{=}} 0.249 \pm 0.003\,\text{g}$

References

[1] Erich Meister. *Praktische Allgemeine Chemie*. 2023.

[2] Fernando Pérez and Brian E. Granger. "IPython: a System for Interactive Scientific Computing". In: *Computing in Science and Engineering* 9.3 (May 2007), pp. 21–29. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53. URL: https://ipython.org.

[3] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2021. URL: https://www.R-project.org/.

[4] *Reinstwassersystem GenPure UV-TOC/UF inkl.Tischdispenser- /Steuereinheit X-CAD*. URL: https://www.huberlab.ch/huberlab/en/CHF/Assortment/Laboratory-consumables-%26-instruments/Water-Purification/Ultrapure-water-purification-systems/-/p/cat_28133. (accessed: 26.01.2024).

[5] Michael Wollensack. *METAS UncLib*. URL: https://www.metas.ch/metas/en/home/fabe/hochfrequenz/unclib.html. (accessed: 12.01.2024).

A Appendix

A.1 R and Python scripts

A.1.1 Correlation between temperature and conductivity

```
# use radian as better R shell

rm(list=ls())

WIDTH <<- 5
HEIGHT <<- 4

# setwd("LFK_Auswertung")
# quartz(height=HEIGHT, width=WIDTH)

library(readxl)

R <<- 8.314462618    # J K^-1 mol^-1
p0 <<- 1013.25

COL_ACET_PRIM <<- "#458B00"
COL_ACET_SEC <<- "#7FFF00"

COL_NHEX_PRIM <<- "#0000CD"
COL_NHEX_SEC <<- "deepskyblue"



IMPORT_PATH <<- "raw_data/"
EXPORT_PATH <<- "exports/"


# HELPER FUNCTIONS
source("../KAL_Auswertung/helpers.R")
#source("kal_routines.R")


temp.data = read_excel("raw_data/LFK_Messprotokolle.xlsx",
↪  sheet="temperatur")

temp = temp.data$"temp (°C)"
lfk = temp.data$"kappa (mS/cm)"



par(mai = c(1,1.1,0.3,0.3))
plot.init.grey(
```

```
  temp,
  lfk,
  xlim=c(24, 55),
  ylim=c(19,30.5),
  xlab=expression(italic(theta)*" / "*"°C"),
  ylab=expression(italic(kappa)*" / "*"mS/cm")
)
points(temp, lfk)

reg = plot.regression(
  temp, lfk,
  slope.annot.x.offset=0.2,
  slope.annot.y.offset=0.1,
  slope.annot=(
    function(slope)(
      TeX(paste(r"(slope:)", sprintf("%0.4f", slope), r"(mS/cm/K)"))
    )
  ),
  delta.annot.x=(function(delta) TeX(paste(r"(\Delta\theta =)",
  ↪  sprintf("%0.2f", delta), "K"))),
  delta.annot.y=(function(delta) TeX(paste(r"(\Delta\kappa =)",
  ↪  sprintf("%0.2f", delta), "mS/cm"))),
)
reg.a = reg$coefficients[1]
reg.b = reg$coefficients[2]

lfk.zero = predict.linear(reg.a, reg.b, 25)

alpha = reg.b / lfk.zero
print(summary(reg))
print(alpha)
print(lfk.zero)


lines(
  c(25,25),
  c(0,lfk.zero),
  lw=2,
)
lines(
  c(0,25),
  c(lfk.zero,lfk.zero),
  lw=2,
)


plot.save(EXPORT_PATH, "lfk_temperature.pdf")
```

A.1.2 Molar conductivity of different electrolytes

```
# use radian as better R shell

rm(list=ls())

WIDTH <<- 5
HEIGHT <<- 4

# setwd("LFK_Auswertung")
# quartz(height=HEIGHT, width=WIDTH)

library(readxl)


IMPORT_PATH <<- "raw_data/"
EXPORT_PATH <<- "exports/"


# HELPER FUNCTIONS
source("../KAL_Auswertung/helpers.R")
#source("kal_routines.R")


temp.data = read_excel("raw_data/LFK_Messprotokolle.xlsx", sheet="molar")
v.start = temp.data$"V (mL)"[1]
lfk.start = temp.data$"kappa (uS/cm)"[1]
temp.data = temp.data[2:nrow(temp.data),]

lfk.brutto = temp.data$"kappa (uS/cm)"
lfk = lfk.brutto - lfk.start

MOLARITY <<- 0.01
CONC.TITER <<- 0.01
v.abs = temp.data$"V (mL)"
v.rel = v.abs - v.start
conc = CONC.TITER * v.rel / v.abs

par(mai = c(1,1.1,0.3,0.3))
par(mgp = c(2.5,0.8,0))
plot.init.grey(
    conc,
    lfk,
    log="xy",
    type="n",
    xlab=expression(italic(c)*" / "*"M"),
    ylab=expression(italic(kappa)*" / "*"mS/cm"),
    xaxt="n", yaxt="n",  # keine Achsenskalen
```

```
    xaxs="i", yaxs="i",  # Achsenbreite exakt
    xlim=c(min(conc) / 1.2, max(conc) * 1.2),
    ylim=c(min(lfk) / 1.2, max(lfk) * 1.2),
)

x.line = c(0.00005, max(conc))
y.line = x.line / 0.00005 * 11.4
lines(
    x.line,
    y.line,
    lty=3,
    lw=2,
    col="black"
)

points(
    conc,
    lfk.brutto,
    pch=21,
    col="#878787",
    bg="#c4c4c4",
)

points(
    conc,
    lfk,
    pch=21,
    col="black",
    bg="white",
)


logticks = plot.axis.log()
axis(1, at=logticks$small, labels=FALSE, tcl=-0.25)
axis(1, at=logticks$big, labels=logticks$big.lab, tcl=-0.25)
axis(2, at=logticks$small, labels=FALSE, tcl=-0.25)
axis(2, at=logticks$big, labels=logticks$big.lab, tcl=-0.25, las=1)

slope = 11.4 / 0.00005  # mS/cm/M
plot.annot(
    0.0006,
    20,
    TeX(paste(r"(slope:)", sprintf("%0.0f", slope), "mS/cm/M"))
)
slope.normalunit = slope / 1000  # S*cm^2/M
print(slope.normalunit)

plot.save(EXPORT_PATH, "lfk_molar_1.pdf")
```

```
conc.sqrt = sqrt(conc)
lfk.molar = lfk / conc / 1000  # S cm^2 / mol
lfk.molar.brutto = lfk.brutto / conc / 1000  # S cm^2 / mol

plot(
    conc.sqrt,
    lfk.molar.brutto,
    xlim=c(0, 0.044),
    ylim=c(0, 320),
    pch=21,
    col="#878787",
    bg="#c4c4c4",
    xlab=expression(italic(sqrt(c))*" / "*"M"^"1/2"),
    ylab=expression(italic(Lambda)*" / "*"S "*"cm"^"2"*"mol"^"-1"),
)
plot.grid()

points(
    conc.sqrt,
    lfk.molar.brutto,
    pch=21,
    col="#878787",
    bg="#c4c4c4",
)

reg = plot.regression(
    conc.sqrt,
    lfk.molar,
    draw.annotation = FALSE,
    abline.lty=1
)
reg.a = reg$coefficients[1]
reg.b = reg$coefficients[2]

points(
    conc.sqrt,
    lfk.molar,
    pch=21,
    col="#000000",
    bg="#ffffff"
)
```

```
plot.annot(
    0.022,
    100,
    join("y = ", sprintf("%0.0f", reg.b), " x + ", sprintf("%0.1f",
    ↪  reg.a)),
    xjust=0.5,
    yjust=0.5,
    adj=0.05
)

plot.save(EXPORT_PATH, "lfk_molar_2.pdf")

print(summary(reg))
```

A.1.3 Conductometric titration

```
rm(list=ls())

library(readxl)

R <<- 8.314462618    # J K^-1 mol^-1
p0 <<- 1013.25

COL_ACET_PRIM <<- "#458B00"
COL_ACET_SEC <<- "#7FFF00"

COL_NHEX_PRIM <<- "#0000CD"
COL_NHEX_SEC <<- "deepskyblue"

WIDTH <<- 5
HEIGHT <<- 4

# quartz(height=HEIGHT, width=WIDTH)


IMPORT_PATH <<- "raw_data/"
EXPORT_PATH <<- "exports/"


# HELPER FUNCTIONS
source("../KAL_Auswertung/helpers.R")
#source("kal_routines.R")


titr.data = read_excel("raw_data/LFK_Messprotokolle.xlsx",
↪  sheet="LFK_titration")

v = titr.data$v_delta
lfk = titr.data$kappa

range.prestep = 1:16
range.poststep = 20:40

# regression
reg.pre = lm(lfk ~ v, subset=range.prestep)
reg.post = lm(lfk ~ v, subset=range.poststep)

reg.a.pre = coef(reg.pre)[1]
reg.b.pre = coef(reg.pre)[2]
reg.a.post = coef(reg.post)[1]
reg.b.post = coef(reg.post)[2]
```

```
v.intercept = (reg.a.pre - reg.a.post) / (reg.b.post - reg.b.pre)

v.ausgl.pre = seq(0.0, v.intercept + 1, length=50)
v.ausgl.post = seq(v.intercept - 1, max(v) + 1, length=50)

lfk.ausgl.pre = predict(reg.pre, list(v=v.ausgl.pre),
↪  interval="confidence")
lfk.ausgl.post = predict(reg.post, list(v=v.ausgl.post),
↪  interval="confidence")

plot.init.grey(
  v, lfk,
  xlim=c(7, 11),
  ylim=c(2.23, 2.57),
  xlab=expression(italic(V)*" / "*"mL"),
  ylab=expression(italic(kappa)*" / "*"mS/cm")
)

matlines(v.ausgl.pre, lfk.ausgl.pre, lty=c(1,1,1), lwd=c(3,1,1),
↪  col="black")
matlines(v.ausgl.post, lfk.ausgl.post, lty=c(1,1,1), lwd=c(3,1,1),
↪  col="black")


points(v, lfk)

points(v[range.prestep], lfk[range.prestep], pch=21, bg="white")
points(v[range.poststep], lfk[range.poststep], pch=21, bg="white")

v.conf = 0.132
lines(c(v.intercept, v.intercept), c(0, 2.28), lwd=2)
lines(c(v.intercept - v.conf, v.intercept - v.conf), c(0, 2.274), lwd=1)
lines(c(v.intercept + v.conf, v.intercept + v.conf), c(0, 2.285), lwd=1)

#plot.save(EXPORT_PATH, "lfk_titration_full.pdf")
plot.save(EXPORT_PATH, "lfk_titration_zoom.pdf")
```

A.1.4 Plotting helper scripts

```
# by github.com/janjoch, 2023

library(latex2exp)

# HELPER FUNCTIONS
join <- function(...) {
  paste(..., sep="")
}

# hack copied from https://stackoverflow.com/questions/1826519/how-to-
↪  assign-from-a-function-which-returns-more-than-one-value
':=' <- function(lhs, rhs) {
  frame <- parent.frame()
  lhs <- as.list(substitute(lhs))
  if (length(lhs) > 1)
    lhs <- lhs[-1]
  if (length(lhs) == 1) {
    do.call(`=`, list(lhs[[1]], rhs), envir=frame)
    return(invisible(NULL))
  }
  if (is.function(rhs) || is(rhs, 'formula'))
    rhs <- list(rhs)
  if (length(lhs) > length(rhs))
    rhs <- c(rhs, rep(list(NULL), length(lhs) - length(rhs)))
  for (i in 1:length(lhs))
    do.call(`=`, list(lhs[[i]], rhs[[i]]), envir=frame)
  return(invisible(NULL))
}

# usage
# func_that_returns_three_values <- function(a,b,c) {
#   return(list(a,b,c))
# }
# c(a,b,c) := func_that_returns_three_values(1,2,3)


predict.linear <- function(reg.a, reg.b, x) {
  y = reg.a + reg.b * x
  y
}


FBy <- function(x, y, sy, ...) {
  # copied from Meister
  arrows(x, y - sy, x, y + sy, code=3, angle=90, length=0.02)
  points(x, y, pch=21, ...)
```

```
}



# PLOT MODULAR FUNCTIONS
plot.init.grey <- function(
  x,
  y,
  xlim=NULL,
  ylim=NULL,
  xlab=expression(italic(t)*" / "*"s"),
  ylab=expression(italic(T)*" / "*"°C"),
  type="l",
  lwd=2,
  col="darkgrey",
  ...
) {
  plot(
    x,
    y,
    type=type,
    lwd=lwd,
    col=col,
    xlim=xlim,
    ylim=ylim,
    xlab=xlab,
    ylab=ylab,
    ...
  )
  plot.grid()
}


plot.grid <- function(nx=NULL, ny=NULL, lty=1, col="lightgray", lwd=1,
↪  ...) {
  grid(
    nx = nx,
    ny = ny,
    lty = lty,      # Grid line type
    col = col, # Grid line color
    lwd = lwd,      # Grid line width
    ...
  )
}

plot.axis.log <- function(begin=1e-10, end=1e10) {
  d = 10^c(-99:99)
  d = d[d>=begin & d<=end]
```

```
  dd = outer(c(1:9), 10^c(-99:99))
  dd = dd[dd>=begin & dd<=end]
  dlab = do.call(
    "expression",
    lapply(seq(along=log10(d)), function(i) substitute(10^E,
    ↪  list(E=log10(d)[i])))
  )
  list("big"=d, "small"=dd, "big.lab"=dlab)
}

plot.line.highlight <- function(x, y, col="blue") {
  lines(
    x,
    y,
    col=col,
    lw=3
  )

}

plot.line.annot <- function(x, y) {
  lines(
    x,
    y,
    lty=3,
    lw=2,
    col="black"
  )

}

plot.annot <- function(x, y, text, xjust=0.5, yjust=0.5, adj=0.15, ...) {
  legend(
    x,
    y,
    text,
    bg="white",
    box.col="white",
    adj=adj,
    xjust=xjust,
    yjust=yjust,
    ...
  )
}

plot.regression <- function(
  x,
  y,
```

```
  draw.annotation=TRUE,
  slope.annot=(function(slope)(TeX(paste(r"(slope:)", sprintf("%0.4f",
  ↪  slope), slope.unit)))),
  slope.annot.x.offset=0,
  slope.annot.y.offset=0,
  abline.lty=2,
  delta.annot.x=(function(delta)(TeX(paste(r"(\Delta t =)",
  ↪  sprintf("%0.2f", delta), "s")))),
  delta.annot.y=(function(delta)(TeX(paste(r"(\Delta T =)",
  ↪  sprintf("%0.2f", delta), "K")))),
  ...
) {
    reg <- lm(y ~ x)
    y.pred <- predict(reg)

    x.min = min(x)
    x.max = max(x)
    y.pred.min = min(y.pred)
    y.pred.max = max(y.pred)

    x.delta = x.max - x.min
    y.pred.delta = y.pred.max - y.pred.min


    if(draw.annotation) {
      # delta t
      plot.line.annot(
        c(min(x), max(x)),
        c(min(y.pred), min(y.pred))
      )
      plot.annot(
        mean(c(min(x), max(x))),
        min(y.pred),
        delta.annot.x(x.delta)
      )

      # delta T
      plot.line.annot(
        c(max(x), max(x)),
        c(min(y.pred), max(y.pred))
      )
      plot.annot(
        max(x),
        mean(c(min(y.pred), max(y.pred))),
        delta.annot.y(y.pred.delta)
      )

      # slope
```

```
      plot.annot(
        min(x) + slope.annot.x.offset * x.delta,
        max(y.pred) - slope.annot.y.offset * y.pred.delta,
        slope.annot(summary(reg)$coef[2,1]),
        xjust=0.3,
        yjust=1
      )
    }

    # Regressionsgerade
    abline(reg, lty=abline.lty, lw=2, ...)

    reg
}

plot.save <- function(export.path, export.plot) {
  dev.copy2pdf(file=join(export.path, export.plot), width=WIDTH,
  ↪  height=HEIGHT)
}
```

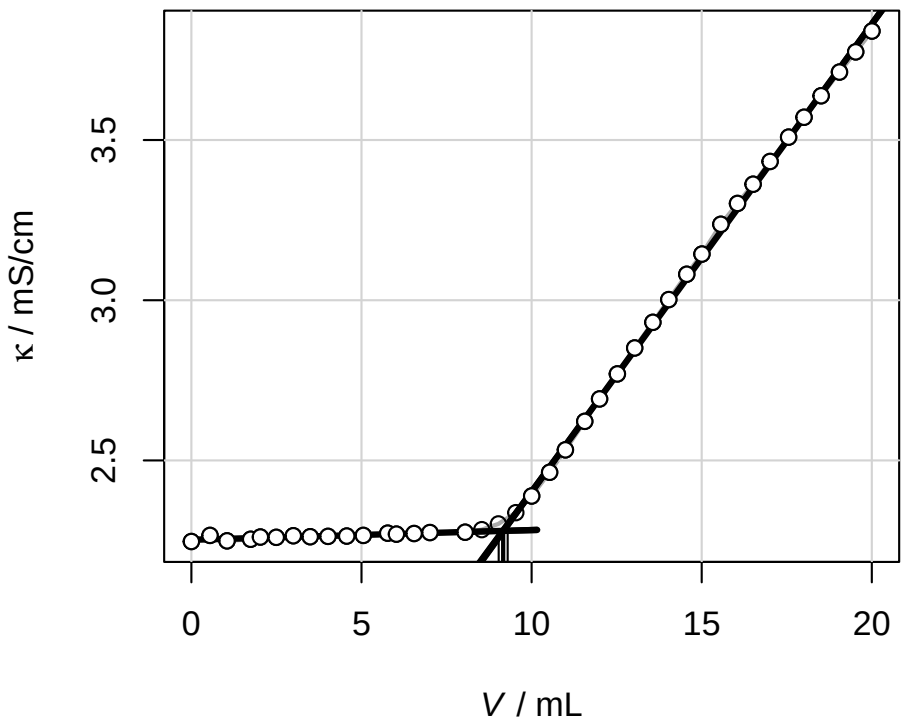

Figure 8: Full view of the titration of K_2CO_3 with 0.099 97 M $CaCl_2$.

A.1.5 Conductivity of differently treated water samples and molar conductivity of different electrolytes

In [9]:
```
import numpy as np

import pandas as pd
```

In [4]:
```
from metas_unclib import *
```

In [6]:
```
water_tap = ufloatfromsamples(
    (289.5, 281.1, 278.5),
    desc="water tap"
)
water_tap
```

Out[6]: 283.03333333333336 ± 7.2867724653006665

In [7]:
```
water_dei= ufloatfromsamples(
    (2.84, 2.9, 2.8),
    desc="water deionized"
)
water_dei
```

Out[7]: 2.8466666666666662 ± 0.0637931056763051

In [8]:
```
water_pure = ufloatfromsamples(
    (1, 1.2, 0.97),
    desc="water purified, degased"
)
water_pure
```

Out[8]: 1.0566666666666666 ± 0.15847230355837524

In [14]:
```
print(pd.DataFrame(np.array(((1,2),(3,4)))).to_latex())
```

```
\begin{tabular}{lrr}
\toprule
{} &  0 &  1 \\
\midrule
0 &  1 &  2 \\
1 &  3 &  4 \\
\bottomrule
\end{tabular}
```

Molar

K2CO3

In [20]:
```
kalium_M = ufloat(138.205, desc="molar mass potassium carbonate")  # g / mo
kalium_m = ufloat(1.3895, 0.0001, desc="mass potassium carbonate")
kalium_c = kalium_m / kalium_M / 0.1 / 10
kalium_c
```

Out[20]: 0.010053905430338989 ± 7.235628233421367e-07

In [24]:
```
kalium_lambda = ufloatfromsamples(
    (2.163, 2.190, 2.188),
    desc="molar conductivity K2CO3"
) / kalium_c
kalium_lambda  # S cm2 / mol
```

Out[24]: 216.86431690056378 ± 1.8966262009802657

Na2CO3

In [25]:
```
natrium_M = ufloat(105.99, desc="molar mass sodium carbonate")  # g / mol
natrium_m = ufloat(1.3895, 0.0001, desc="mass sodium carbonate")
natrium_c = natrium_m / natrium_M / 0.1 / 10
natrium_c
```

Out[25]: 0.01310972733276724 ± 9.434852344560808e-07

In [26]:
```
natrium_lambda = ufloatfromsamples(
    (1.715, 1.720, 1.720),
    desc="molar conductivity Na2CO3"
) / natrium_c
natrium_lambda  # S cm2 / mol
```

Out[26]: 131.073155811443 ± 0.2792487421399839

Exercises from the book

$$v_i = \frac{e E}{6 \pi \eta} \frac{z_i}{r_i}$$

Ex 1

In [28]:
```
from math import pi
```

In [33]:
```
e = 1.602176e-19  # C
E = 100  # V / m
eta = 1e-3  # Pa s
r = 1e-9  # m

z_mg = 2
z_essig = 1
```

In [34]:
```
e * E / (6 * pi * eta) * z_mg / r  # m/s
```

Out[34]: 1.6999615340213366e-06

In [35]:
```
e * E / (6 * pi * eta) * z_essig / r  # m/s
```

Out[35]: 8.499807670106683e-07