```
 1: # quotienten-pyrometry bei spektrum
 2: # 7.6.2020
 3: # by rs
 4:
 5: rm(list = ls())
 6:
 7: path <-dirname(rstudioapi::getActiveDocumentContext()$path)
 8: setwd(path)
 9:
10: # Parameters
11: real_curve = FALSE
12: real_curve = TRUE
13: data_file = "data.borchert/Metall_2s_1kW_nStelle_2_400.dat"
14: calib_messung_file = "data.borchert/B12A_Messung.txt"
15: calib_factor_file = "data.borchert/B12Aacalw.txt"
16: calib_emiss <- function(lambda,T) {
17:   eps_b0=0.5296; eps_b1=5.9531e-6; eps_b2=3.7677e-9
18:   eps_b0 - lambda * 10 * (eps_b1 + eps_b2*T)
19: }
20: T_cal = 2448  # in K
21:
22: d_lambda = 5  # Abstand in nm
23: T_range = seq(200,4000,10)
24: noise = 100
25: smooth_fac = 10000
26: lambda_restrict = c(100,1200)
27:
28: #------------------------
29:
30: source(file="quotienten-pyrometry bei spektrum - includes.r") # Definition of
    functions
31:
32: #------------------------
33: # get data from measurement
34:
35:   if (real_curve) {
36:     # get measurement data
37:     data.in <- read.table(data_file, header = FALSE, col.names=c("lambda","L"),
38:       skipNul=TRUE, na.strings ="NaN",dec=".",sep=",")
39:     data.in$lambda = data.in$lambda/10
40:     # plot(data.in,type="l")
41:     lambda_range = data.in$lambda
42:     spec_data = data.in
43:
44:     # get calibration measurement data
45:     calib_messung.in <- read.table(calib_messung_file, header = FALSE,
    col.names=c("lambda","L"), skipNul=TRUE,
46:                                    na.strings ="NaN",dec=",",sep="\t",
    comment.char=">")
47:     calib_messung = approx(calib_messung.in$lambda, calib_messung.in$L-0,
    xout=lambda_range)
48:     calib_messung = data.frame(lambda=calib_messung$x, L=calib_messung$y)
49:     # plot(calib_messung,type="l")
50:     spec_data = calib_messung
51:
52:     # get emissivity for calibration
53:     calib_emiss.in = data.frame(lambda=lambda_range,
    emiss=calib_emiss(lambda_range,T_cal))
54:     # plot(calib_emiss.in,type="l")
55:     calib_emiss_func = approxfun(calib_emiss.in$lambda, calib_emiss.in$emiss)
56:     # plot(lambda_range, calib_emiss_func(lambda_range),type="l")
57:
58:     # get calibration curve from file
59:     calib_func.in <- read.table(calib_factor_file, header = FALSE,
    col.names=c("lambda","factor"), skipNul=TRUE,
```

```
60:                                   na.strings ="NaN",dec=".",sep="",
    comment.char=">")
61:     calib_func.in$lambda = calib_func.in$lambda/10
62:     calib_func = approxfun(calib_func.in$lambda, calib_func.in$factor, ties=mean)
63:     # plot(lambda_range, calib_func(lambda_range),type="l")
64:   }
65: # windows()
66: # dev.list()["RStudioGD"]
67: # dev.set(dev.list()["RStudioGD"])
68:
69: #------------------------
70: # make artifical data
71:
72:   if (!real_curve) {
73:     lambda_range=seq(200,3000,1)
74:     calib_emiss_func = approxfun(c(300,600,10000),c(0.8,0.5,0.2), rule=2)
75:     calib_emiss_func = cobs(lambda_range,calib_emiss_func(lambda_range),
    lambda=10000)
76:     calib_emiss_func = approxfun(predict(calib_emiss_func,lambda_range))
77:     # plot(calib_emiss_func(lambda_range), type="l")
78:
79:     calib_knots = list(x=c(100,300,500,1000,3500), y=c(1e-5,1e-5,1e-3,1e-5,1e-13))
80:     calib_sensibility_func = approxfun(calib_knots$x, calib_knots$y, rule=2)
81:     calib_sensibility_func = cobs(lambda_range,
    calib_sensibility_func(lambda_range),lambda=0, degree=2)
82:     calib_sensibility_func = approxfun(predict(calib_sensibility_func,
    lambda_range))
83:     plot(calib_sensibility_func(lambda_range), type="l")
84:
85:     spec_data = spec_data_make(lambda_range, plot=TRUE, eps=calib_emiss_func,
    sensibility=calib_sensibility_func, T=T_cal, noise=noise)
86:     calib_messung = spec_data
87:   }
88:
89: #------------------------
90: # get lambda-range
91:   lambda_range = range(spec_data$lambda)
92:   lambda_range = c(max(lambda_range[1],lambda_restrict[1]), min(lambda_range[2],
    lambda_restrict[2]))
93:   lambda_range = seq(lambda_range[1], lambda_range[2], d_lambda)
94:
95: #------------------------
96: # calc reference black body matrix for T_range and lambda_range with radiation
    and ratio from adjacent wavelengths
97:
98:   bb = black_body_ref(T_range, lambda_range, plot=TRUE)
99:   bb_L_group_T = group_by(bb$L,T)
100:  bb_L_group_lambda = group_by(bb$L,lambda)
101:
102: #------------------------
103: # smooth calibrate data, adapt to lambda_range
104:   # spec_calib_messung_pre = smooth_curve(calib_messung, lambda=lambda_range,
    plot=TRUE, span=smooth_fac)
105:   title_y = eval(bquote(expression("L("*lambda*",T="*.(T_cal)*"K)") ))
106:   spec_calib_messung_pre = smooth_curve(calib_messung, lambda=lambda_range,
    title_y=title_y, plot=TRUE, fac=smooth_fac)
107:   colnames(spec_calib_messung_pre) = c("lambda", "L")
108:   spec_calib = spec_calib_messung_pre
109:   # plot(spec_calib, type = "l")
110:
111:   # normalize to black body and calc normalisation factor
112:   spec_calib$L = spec_calib$L/calib_emiss_func(lambda_range)
113:   bb_ref = bb_calc(lambda_range, T_cal)
114:   # plot(bb_ref, type = "l")
```

```
115:   calib_norm = spec_calib$L/bb_ref$L
116:   # plot(calib_norm, type = "l")
117:
118: #------------------------
119: # smooth data and adapt to lambda_range, calc quotient from adjacent wavelengths
120:
121:   spec_data_pre = smooth_curve(spec_data, lambda=lambda_range, plot=TRUE,
     fac=smooth_fac)
122:   colnames(spec_data_pre) = c("lambda", "L")
123:
124:   spec_data_pre2 = spec_data_pre
125:   spec_data_pre2$L = spec_data_pre$L/calib_norm
126:   # plot(spec_data_pre2, type = "l")
127:
128:   spec_ratio = spec_data_ratio(spec_data_pre2,lambda=lambda_range, plot=TRUE)
129:   # plot(spec_ratio, type = "l")
130:
131: #------------------------
132: # look for equivalent temperature
133:
134:   # bb_L_group_lambda %$% plot(T~lambda) # operator aus magrittr
135:
136:   T_find <- function(lambda_in, value, bb_) {
137:     bb_vals <- filter(bb_, near(lambda, lambda_in, tol =
     2000*.Machine$double.eps))
138:     T = approx(bb_vals$bb_ratio,bb_vals$T, value)$y
139:   }
140:
141:   T_res = list()
142:   for (xx in 1:length(spec_ratio$lambda)) {
143:     T_res=rbind(T_res, c(lambda=spec_ratio$lambda[xx],
     T_res=T_find(spec_ratio$lambda[xx], spec_ratio$ratio[xx], bb$ratio)))
144:   }
145:   T_res
146:   plot(T_res, type = "l")
147:
148:   sapply(spec_ratio, function(X) {
149:     print("a")
150:     print(X)
151:   })
152:
153:   replicate(2,spec_ratio, function(X) {
154:     print("a")
155:     print(X)
156:   })
157:
158:   # # folgendes geht nicht, bb wird nur einmal als ganzes verwertet
159:   # T_find2 <- function(value, bb_vals) {
160:   #   print(value)
161:   #   print(bb_vals)
162:   #   T = approx(bb_vals$bb_ratio,bb_vals$T, value)$y
163:   # }
164:   #
165:   # T_res2 = mutate(spec_ratio, T=T_find2(ratio,
     filter(bb$ratio[near(bb$ratio$lambda, lambda, tol = 2000*.Machine$double.eps),])
     ) )
166:   #
167:   # aa =  bb$ratio[near(bb$ratio$lambda, 740.88, tol = 2000*.Machine$double.eps),
168:
169:
170: # group_map(bb_group_lambda, ~min(.x$T))
171: #
172: # # ----------------------------
173: # #
```

```
174: # # library(purrr)
175: #
176: # select(L_ref_ratio_group, lambda)
177: # L_ref=filter(L_ref_ratio_group, lambda == 0.000000325)
178: # filter(L_ref_ratio_group, T==200)
179: # approx(L_ref$value,L_ref$T,0.85)
180: #
181: # filter(L_ref_ratio_group, map(L_ref_ratio_group,
     ~any(near(L_ref_ratio_group$lambda, 6.75e-07, tol = .Machine$double.eps^0.5)))[1]
     )
182: # L_ref_ratio_group[1,]
183: #
184: # filter(map_lgl(x, ~ any(near(.x, c(0.5679, 5.6789), tol = 1e-4))))
185: #
186: #
187: #
188: # # ----------------------------
189: # # Messung
190: #
191: # T1 = 3463
192: # T2 = 3360
193: # sigma = 5.670400e-12 # Stefan-Boltzmann constant in W/(cm^2*K^4)
194: # q = sigma * (T1^4 - T2^4) # radiant emittance
195: # q
196: #
197: #
198: # # ************************************************************************
199: #
200: # # Moeglichkeiten zum Verbessern
201: #
202: # L = expression(2*h*c^2/lambda^5*1/(exp(h*c/(lambda*k*T))-1));
203: # deriv(L_, "lambda");
204: #
205: # library("Deriv")
206: # fx = Deriv("sin(x^2) * y", "x")
207: # fx_ = as.formula(paste("y ~ ", fx))
208: # eval(fx_)
209: #
210: # deriv(~ x^2, "x")
211: # dx2x <- deriv(~ x^2, "x") ; dx2x
212: # mode(dx2x)
213: # eval(dx2x)
214: # dx2x.gradient
215: #
216: # L = "2*h*c^2/lambda^5*1/(exp(h*c/(lambda*k*T))-1)"
217: # fx = Deriv(L, "lambda")
218: #
219: # deriv(~ 2*h*c^2/lambda^5*1/(exp(h*c/(lambda*k*T))-1), "lambda")
220:
```