Estimating Chi - examples

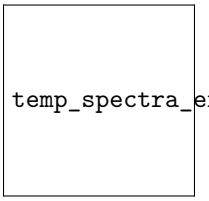
This notebook imports the temperature spectral data saved from matlab (from structure `diss.sclar_spectra.scalar_spec`), corrects for the frequency response of the sensor using Vachon and Lueck correction (used by Rockland in odas) and fits the model stated in Bluteau to the temperature spectrum to estimate χ.

Data from profile_nr 32 was used for testing, dive depth was 414 m. 146 spectra were calculated.

Correcting for frequency response

```
# correction as in odas library
# time constant for double-pole response:
F_0 <- 25*sqrt(mean_speed)
tau_therm <- 2*pi*F_0 / sqrt(sqrt(2) - 1)
tau_therm <- 1 / tau_therm
# correction:
# create a NA matrix for gradT1 and T2 of correct size and then fill with corrected values
gradT1_c <- matrix(data=NA, nrow=length(freq[,1]), ncol=length(freq[1,]))
gradT2_c <- matrix(data=NA, nrow=length(freq[,1]), ncol=length(freq[1,]))
for(index in 1:length(freq[1,])){
  gradT1_c[,index] <- gradT1[,index] * (1 + (2*pi*tau_therm[index]*freq[,index])^2)^2
  gradT2_c[,index] <- gradT2[,index] * (1 + (2*pi*tau_therm[index]*freq[,index])^2)^2
} # end for loop
```

Plot corrected sample spectra



temp_spectra_examples_files/figure-latex/unnamed-chunk-4-1.pdf

Fit model to spectra with χ as parameter estimated with MLE (maximum likelihood estimation)

model to fit:

$$\Psi_{\delta T/\delta x_i}(k) = C_T \chi \epsilon^{-1/3} k^{1/3}$$

with:

- $\Psi_{\delta T/\delta x_i}$ = corrected energy spectrum (`gradT_c`),
- C_T = Obukhov-Corrsin universal constant with values between 0.3 and 0.5, 0.4 recommended,
- ϵ = dissipation energy estimated from shear probe signal (`eps1`,`eps2`),
- k = wavenumber (k is here in rad/m, a conversion coefficient to convert to cpm should be added to the model?) (`waven`)

Convert measured spectra to wavenumber spectra by multiplying with mean_speed

```
# make empty (NA) matrix of correct size
P_gradT1_c <- matrix(NA, nrow = length(gradT1_c[,1]), ncol=length(gradT1_c[1,]))
P_gradT2_c <- matrix(NA, nrow = length(gradT2_c[,1]), ncol=length(gradT2_c[1,]))
# fill matrix with converted values
for(index in 1:length(gradT1_c[,1])){
```

```
  P_gradT1_c[index,] <- gradT1_c[index,] * mean_speed
  P_gradT2_c[index,] <- gradT2_c[index,] * mean_speed
}
```

Find upper limit for k for model fit using the criterium correction $H(k) \leq 3$ and $k \leq 0.1\eta^{-1}$

```
# initialise K_max
K_max <- c()
# loop through all spectra in profile 32
for(segment in 1:146){
  # calculate correction factor Hf
  f <- freq[,segment]
  tau0 <- 4.1 * 10^-3
  speed0 <- 1
  tau <- tau0 * (mean_speed[segment,]/speed0)^(-0.5)
  Hf <- (1 + (2 * pi * tau *f)^2)^(2)
  # find range where Hf does not exceed 3
  ind <- length(Hf[Hf <= 3])
  # extract K for range where Hf <= 3
  K_max_seg = waven[ind,segment]

  # calculate eta (Kolmogorov length scale) from epsilon for check of second criterium for upper k
  eta <- (((10^-6)^3)/mean(eps1[segment],eps2[segment]))^0.25 # Kolmogorov length scale

  # final upper limit of k
  K_max[segment] <- min(0.1/eta,K_max_seg)
}
```

Define x and y for model, only use spectrum for wavenumbers < K_max

```
# initialise x and y as empty lists
x <- list()
y1 <- list()
y2 <- list()
# loop through all spectra for profile 32
for(segment in 1:146){
  K_min <- 0 # K_min needs to be checked after fitting, default value = 0
  # set lower k limit index
  min_ind <- max(length(waven[waven[,segment]<=K_min, segment]),2)
  # set upper k limit index
  max_ind <- length(waven[waven[,segment]<=K_max[segment],segment])
  # define y, y values are corrected wavenumber temp-gradient spectrum --> P_gradT1_c for chosen k rang
  y1_seg <- P_gradT1_c[min_ind:max_ind,segment]
  y2_seg <- P_gradT2_c[min_ind:max_ind,segment]
  # define x, x values are k (wavenumbers) for chosen range
  x_seg <- waven[min_ind:max_ind,segment]
  # append x and y lists in each step
  y1 <- c(y1,list(y1_seg))
  y2 <- c(y2,list(y2_seg))
  x <- c(x,list(x_seg))
} # end for loop
# combine y1 and y2 in one list
y <- list(y1, y2)
```

maximum k and number of spectral points included in estimating χ for sample spectra:

- spectrum 16: $k_{max} = 29.5$, n = 74
- spectrum 30: $k_{max} = 18.3$, n = 44
- spectrum 50: $k_{max} = 15.6$, n = 37
- spectrum 70: $k_{max} = 22.1$, n = 50
- spectrum 100: $k_{max} = 18.5$, n = 39

definition of maximum likelihood function

```
# define C_T
C_T <- 0.4
# initialise output fit and chi
fit <- list()
chi <- c()
# loop through both temp sensors
for(sensor in 1:2){
  fit_sens <- list()
  chi_sens <- c()
  # loop through each spectrum in profile 32
  for(segment in 1:146){
    xs <- x[[segment]]
    ys <- y[[sensor]][[segment]]
    #likelihood function definition
    LL <- function(chi) {
      # Find residuals
      R = ys - C_T * chi * mean(eps1[segment],eps2[segment])^(-1/3) * xs^(1/3)
      # Calculate the likelihood for the residuals
      R = suppressWarnings(dnorm(R, log = TRUE))
      # Sum the log likelihoods for all of the data points
      -sum(R)
    } # end of LL function

    #model fit
    fit_seg <- suppressWarnings(mle(LL, start = list( chi=10^(-7))
                                ,nobs = length(ys),  method='L-BFGS-B'))
    fit_sens <- c(fit_sens, list(fit_seg))
    chi_sens <- c(chi_sens, fit_seg@coef[[1]])
  } # end for loop through spectra
  fit <- c(fit, list(fit_sens))
  chi <- list(chi, chi_sens)
} # end for loop through sensors
```

Estimated χ for sample spectra

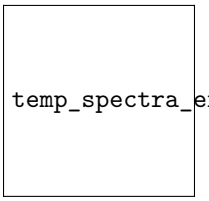
```
## [1] "spectrum 16: estimated Chi = 3.25e-07, Eps = 8.92e-09, logLik = -68, AIC = 138"
## [1] "spectrum 30: estimated Chi = 4e-08, Eps = 1.13e-09, logLik = -40.4, AIC = 82.9"
## [1] "spectrum 50: estimated Chi = 3.64e-07, Eps = 5.98e-10, logLik = -34, AIC = 70"
## [1] "spectrum 70: estimated Chi = 6.57e-07, Eps = 2.37e-09, logLik = -45.9, AIC = 93.9"
## [1] "spectrum 100: estimated Chi = 3.07e-07, Eps = 1.18e-09, logLik = -35.8, AIC = 73.7"
## [1] "spectrum 130: estimated Chi = 4.41e-08, Eps = 3.52e-10, logLik = -25.7, AIC = 53.5"
```

calculate model with fitted χ for plotting

```
spec_model <- list()
for(sensor in 1:2){
  spec_model_sens <- list()
  for(segment in 1:146){
    spec_model_seg <-  C_T * fit[[sensor]][[segment]]@coef[1] * mean(eps1[segment],eps2[segment])^(-1/3)
    spec_model_sens <- c(spec_model_sens, list(spec_model_seg))
  }#end for loop through spectra
  spec_model <- c(spec_model, list(spec_model_sens))
}#end for loop through sensors
```

plot data and model for sample spectra

```
pl_model <- data.frame()
for(segment in specnr_sub){
  k_model_s <- x[[segment]]
  spec_model1_s <- spec_model[[1]][[segment]]
  spec_model2_s <- spec_model[[2]][[segment]]
  pl_model <- pl_model %>%
      rbind(data.frame(k_model = k_model_s, T1 = spec_model1_s, T2 = spec_model2_s, specnr = as.factor(s
    gather(key = sensor, -k_model, -specnr, value = spec_model))
} #end for loop through spectra
pl_data <- gradT_df %>%
  filter(specnr %in% specnr_sub, freq != 0) %>%
  mutate(specnr = as.factor(specnr))
ggplot() +
  geom_point(data=pl_data, aes(x=freq, y = gradT_c, col = sensor), alpha = 0.3) +
  geom_line(data=pl_model, aes(x=k_model, y=spec_model, col = sensor)) +
  scale_y_continuous(trans='log10', limits=c(10^-9,10^-2)) +
  scale_x_continuous(trans='log10')  +
  labs(x = 'freq', y = 'spectra energy') +
  facet_wrap(~specnr) +
  theme_bw()
```



temp_spectra_examples_files/figure-latex/unnamed-chunk-11-1.pdf

check for lower k condition using MAD (mean absolute deviation) between observed and modelled spectra

```
# only checking T1 for now
MAD <- list()
crit <- list()
spec_good <- c()
  for(segment in 1:146){
    phi <- y[[1]][[segment]]   # corrected wavenumber temp-gradient = measured spectrum
    psi <- spec_model[[1]][[segment]]   # modelled spectrum
    steps <- 6  # 0.5-0.7 of a decade long?
    nr_steps <- floor(length(phi)/steps)
    modulo <- length(phi) %% steps
    if(modulo >= 4){
```

```
    nr_steps <- nr_steps + 1
  }

  # calculate MAD for steps of 6 values for the spectrum
  MAD_seg <- c()
  crit_seg <- c()
  for(i in 1:nr_steps){
    n_st <- (i-1) * steps + 1
    if(i == nr_steps){
      if(modulo >= 4){
        n_en <- n_st + modulo - 1
      }else{
        n_en <- n_st + steps - 1 + modulo
      }
    }else{
      n_en <- n_st + steps - 1
    }
    phi_n = phi[n_st:n_en]
    psi_n = psi[n_st:n_en]
    MAD_seg[i] <-  1/(length(phi_n)) * sum(abs(phi_n/psi_n - mean(phi_n/psi_n)))

    # check whether MAD fits criteria, crit = 0 (bad), 1 (good)
    dof <- length(phi_n) - 1   # degrees of freedom
    crit_seg[i] <- ifelse(MAD_seg[i] < 2 * (2/dof)^0.5, 1, 0)

  } # end for loop
  MAD <- c(MAD, list(MAD_seg))
  crit <- c(crit, list(crit_seg))
  spec_good <- c(spec_good, ifelse(sum(crit_seg)==length(crit_seg),1,0))
}
#If all subsets of the spectrum yielded a MAD>2(2/d)^1/2 (d = degrees of freedom   --> n-1 ?) (Ruddic
```

There are 70 good spectra out of 146 spectra in the profile.

For our sample spectra:

```
## [1] "spectrum 16: all MADs fulfill criteria, good spectrum"
## [1] "spectrum 30: at least one MAD value outside allowed range, change minimum k"
## [1] "spectrum 50: at least one MAD value outside allowed range, change minimum k"
## [1] "spectrum 70: all MADs fulfill criteria, good spectrum"
## [1] "spectrum 100: all MADs fulfill criteria, good spectrum"
## [1] "spectrum 130: all MADs fulfill criteria, good spectrum"
```