Worked example II - modelling river turbulence

Michael Dietze

December 20, 2017

This document is part of the supplementary materials to the article by Dietze (2017).

Introduction

Rivers cause seismic signals due to transported particles impacting the bed, waves at the water surface, cavitation and turbulent flow. For the latter, dominant source, Gimbert et al. (2014) have developed a physical model. This model has been translated to R by Sophie Lagarde and became a part of the 'eseis' package. It calculates a seismic power spectral density estimate based on parameters describing fluid, topography and bedrock/sediment properties as well as a series of seismic boundary conditions. Here the model is applied in a Monte Carlo environment to account for the uncertainty inherent to the input parameters. It is used to generate a set of modelled seismic spectra for different water stages and identify those seismic spectra that best match spectra from an empiric data set, recorded by a TC120s seismometer close to the river during a small flood. The signals were logged at 400 Hz with a Nanometrics Centaur.

The raw seismic data used in this document is too large to be provided along with this repository. It is available from the author upon request. The water level data were not generated by the author but the contact to the provider can be delived upon request. The seismic spectra that were generated from the raw seismic data data are provided as `data_example_II.rda`.

Analysis script preparation

After all necessary packages are loaded and the working directory is set.

```
## load packages
library(eseis)
library(magrittr)
library(rdwd)

## set working directory
setwd(dir = "~/Documents/projects/Environmental_seismology/2017_Wernersbach/")
```

Session information

For a full picture of the R session in which the analysis takes plase, all relevant information is displayed.

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.3 LTS
##
## Matrix products: default
## BLAS: /usr/lib/libblas/libblas.so.3.6.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
```

```
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] rdwd_0.9.0   magrittr_1.5 eseis_0.4.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.14          XML_3.98-1.9          digest_0.6.12
##  [4] rprojroot_1.2         berryFunctions_1.16.3 backports_1.1.1
##  [7] evaluate_0.10.1       pbapply_1.3-3         stringi_1.1.6
## [10] IRISSeismic_1.4.6     rmarkdown_1.8         tools_3.4.3
## [13] stringr_1.2.0         parallel_3.4.3        abind_1.4-5
## [16] yaml_2.1.15           compiler_3.4.3        htmltools_0.3.6
## [19] knitr_1.17
```

Parameter settings

All relevant parameters are defined. The empiric data are clipped to hourly (`t_dur`) intervals, to which later a two-sided buffer (`buffer`) is added to account for processing effects. Based on the duration definition, a sequence of start times (`t`) is generated that describes the small flood over 39 hours. The filter frequencies (`f_filter`) are set to remove contaminating frequency content. The later fit of modelled and empiric data is restricted to a fit frequency window (`f_fit`) of 10-100 Hz. A sequence of water levels (`h`) is defined to generate corresponding model spectra. The Monte Carlo environment is defined by the number of runs (`n`), that should be increased beyond 100 for reliable results. The uncertainty ranges of the turbulence model parameters (`params`) are defined, as well. Finally, the independent water level data set (`w`) is loaded, its time stamp converted to POSIXct standard and the data are clipped to the time period of interest (`t`).

```
## set working directory
setwd(dir = "~/Documents/projects/Environmental_seismology/2017_Wernersbach/")

## set time window size to cut time series
t_dur <-  3600

## define time series of interest
t <- seq(from = as.POSIXct(x = "2017-07-26 09:00:00", tz = "UTC"),
         to = as.POSIXct(x = "2017-07-29 00:00:00", tz = "UTC"),
         by = t_dur)

## set buffer size
buffer <- 300

## set filter frequencies
f_filter <- c(5, 190)

## set fit frequencies
f_fit <- c(10, 100)
```

```
## define water levels to model
h <- seq(from = 0.01,
         to = 2.00,
         by = 0.05)

## define number of Monte Carlo runs
n <- 100

## define spectra model parameter ranges
params <- list(d_s = c(0.001, 0.01),
               s_s = c(1.0, 1.5),
               r_s = c(2600, 2700),
               w_w = c(1.0, 1.5),
               a_w = c(0.03, 0.05),
               r_0 = 1.5,
               f_0 = 1,
               q_0 = c(5, 10),
               v_0 = c(800, 1000),
               p_0 = c(0.40, 0.60),
               n_0 = c(0.5, 0.7, 0.7, 0.9))

## load water level data
w <- read.table(file = "data/water/sfm_based_stage_wernersbach.txt.csv",
                header = TRUE,
                stringsAsFactors = FALSE,
                sep = "\t")

## convert water level time vector to POSIXct format
w$Zeit <- as.POSIXct(x = w$Zeit, tz = "CET")

## truncate data set to period of interest
w <- w[w$Zeit >= min(t) & w$Zeit <= max(t),]
```

```
## Warning in check_tzones(e1, e2): 'tzone' attributes are inconsistent

## Warning in check_tzones(e1, e2): 'tzone' attributes are inconsistent
```

Weather data import

To inspect the model output for signal contamination by meteorological sources (i.e., precipitation), hourly data from the closest meteorological station operated by the DWD, German Weather Bureau, is imported using the R package `rdwd` (Boessenkool, 2017) and clipped to the time period of interest (`t`).

```
## get weather data from closest station (Willsdruff, ID 13654)
link <- selectDWD(id = 13654,
                  res = "hourly",
                  var = "precipitation",
                  per = "recent")

file <- dataDWD(file = link,
                read = FALSE,
                dir = tempdir(),
                quiet = TRUE)
```

```
clim <- readDWD(file = file)

clim <- clim[clim$MESS_DATUM >= min(t) & clim$MESS_DATUM <= max(t),]
```

```
## Warning in check_tzones(e1, e2): 'tzone' attributes are inconsistent

## Warning in check_tzones(e1, e2): 'tzone' attributes are inconsistent
```

Calculate empiric spectra

For each of the hourly time segments (`t`), including the two-sided buffer, the vertical seismic component is imported as `eseis` object and assigned to the data list object `s`. For all successful imports, the signals are deconvolved, filtered (`f_filter`), clipped to remove the buffers, and a spectrum is calculated. The spectra are clipped to meet the defined fit frequency limits (`f_fit`) and returned as lists of data frames containing the frequencies and spectral power values.

```
## define ouput object
s <- vector(mode = "list",
            length = length(t))

## loop through all hourly snippets
for(i in 1:length(s)) {

  ## read seismic file of hour of interest
  s_i <- try(eseis::aux_getevent(start = t[i] - buffer,
                                 duration = t_dur + buffer,
                                 station = "WBTW1",
                                 component = "BHZ",
                                 dir = "~/Documents/projects/Environmental_seismology/2017_Wernersbach/d

  if(class(s_i) != "try-error") {

    ## prepare data and calculate spectrum
    s_i <- eseis::signal_deconvolve(data = s_i, logger = "Centaur") %>%
      eseis::signal_detrend() %>%
      eseis::signal_filter(f = f_filter) %>%
      eseis::signal_clip(limits = c(t[i], t[i] + t_dur)) %>%
      eseis::signal_spectrum(method = "autoregressive")

    s[[i]] <- s_i

    print(paste(i, "OK"))
  } else {

    print(paste(i, "failed"))
  }

}
```

```
## [1] "1 OK"
## [1] "2 OK"
## [1] "3 OK"
## [1] "4 OK"
```

```
## [1] "5 OK"
## [1] "6 OK"
## [1] "7 OK"
## [1] "8 OK"
## [1] "9 OK"
## [1] "10 OK"
## [1] "11 OK"
## [1] "12 OK"
## [1] "13 OK"
## [1] "14 OK"
## [1] "15 OK"
## [1] "16 OK"
## [1] "17 OK"
## [1] "18 OK"
## [1] "19 OK"
## [1] "20 OK"
## [1] "21 OK"
## [1] "22 OK"
## [1] "23 OK"
## [1] "24 OK"
## [1] "25 OK"
## [1] "26 OK"
## [1] "27 OK"
## [1] "28 OK"
## [1] "29 OK"
## [1] "30 OK"
## [1] "31 OK"
## [1] "32 OK"
## [1] "33 OK"
## [1] "34 OK"
## [1] "35 OK"
## [1] "36 OK"
## [1] "37 OK"
## [1] "38 OK"
## [1] "39 OK"
## [1] "40 OK"
## [1] "41 OK"
## [1] "42 OK"
## [1] "43 OK"
## [1] "44 OK"
## [1] "45 OK"
## [1] "46 OK"
## [1] "47 OK"
## [1] "48 OK"
## [1] "49 OK"
## [1] "50 OK"
## [1] "51 OK"
## [1] "52 OK"
## [1] "53 OK"
## [1] "54 OK"
## [1] "55 OK"
## [1] "56 OK"
## [1] "57 OK"
## [1] "58 OK"
```

```
## [1] "59 OK"
## [1] "60 OK"
## [1] "61 OK"
## [1] "62 OK"
## [1] "63 OK"
## [1] "64 OK"
```

```
## get index of frequencies in range of interest
f_ok <- s[[1]]$spectrum$frequency > f_fit[1] &
  s[[1]]$spectrum$frequency < f_fit[2]

## truncate data sets to frequencies of interest
s_trunc <- lapply(X = s, FUN = function(s, f_ok) {

  f_clip <- s$spectrum$frequency[f_ok]
  s_clip <- s$spectrum$spectrum[f_ok]

  return(data.frame(f = f_clip,
                    s = s_clip))
}, f_ok)

## extract frequency vector
f <- s_trunc[[1]]$f
```

Generate modelled spectra in Monte Carlo environment

For each water stage (`h`), the model parameter list (`params`) is used to generate `n` random values to model equally likely power spectra for the given water level. All resulting spectra are coerced to a matrix and the values are converted to dB units. A corresponding water level vector is generated.

```
## define output data set
P <- vector(mode = "list",
            length = length(h))

## run MCMC approach for each water stage value
for(i in 1:length(h)) {

  P_i <- lapply(X = 1:n, FUN = function(n, params, h, f){

    m <- model_turbulence(d_s = runif(1, params$d_s[1], params$d_s[2]),
                          s_s = runif(1, params$s_s[1], params$s_s[2]),
                          r_s = runif(1, params$r_s[1], params$r_s[2]),
                          h_w = h,
                          w_w = runif(1, params$w_w[1], params$w_w[2]),
                          a_w = runif(1, params$a_w[1], params$a_w[2]),
                          f = f,
                          r_0 = params$r_0,
                          f_0 = params$f_0,
                          q_0 = runif(1, params$q_0[1], params$q_0[2]),
                          v_0 = runif(1, params$v_0[1], params$v_0[2]),
                          p_0 = runif(1, params$p_0[1], params$p_0[2]),
                          n_0 = c(runif(1, params$n_0[1], params$n_0[2]),
                                  runif(1, params$n_0[3], params$n_0[4])))
```

```
    return(m$spectrum)

  }, params, h = h[i], f)

  P[[i]] <- do.call(rbind, P_i)
}

## convert list to matrix
P <- do.call(rbind, P)

## convert power to dB
P <- 10 * log10(P)

## assign water level for each MCMC result
h_P <- rep(h, each = n)
```

Compare empiric with modelled spectra

All empiric spectra are compared with the lookup table of modelled spectra (`P`) and only those modelled spectra are kept that have an average absolute difference below the 0.05 quantile of all trials. From these remaining models, the corresponding water levels are used to calculate mean (`h_mean`) and standard deviations (`h_sd`) for later use.

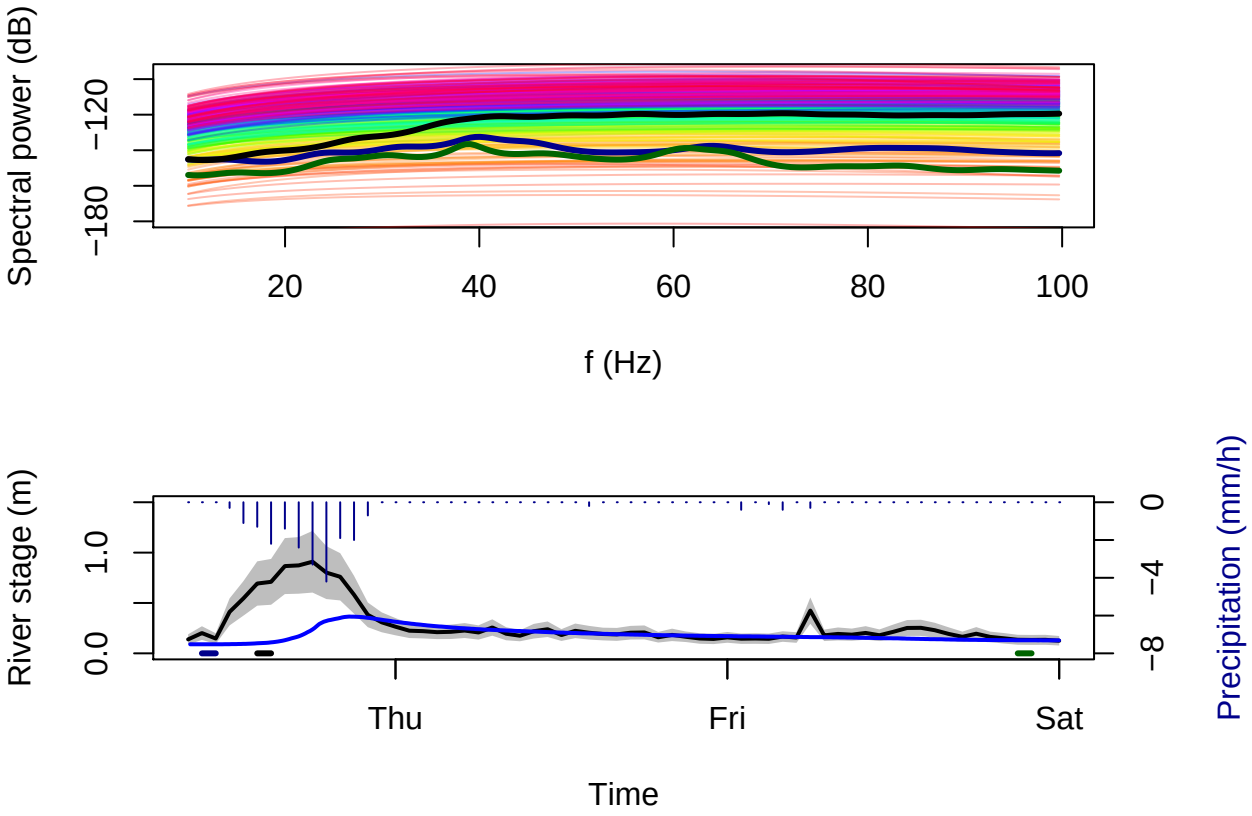
```
## define output variables
h_mean <- numeric(length = length(s_trunc))
h_sd <- numeric(length = length(s_trunc))

## calculate summary statistics for water stages
for(i in 1:length(s_trunc)) {

  ## calculate average absolute difference between empiric and modelled data
  d_s <- rowMeans(abs(P - 10 * log10(s_trunc[[i]]$s)))

  ## isolate best 5 % of fits
  h_ok <- h_P[d_s < quantile(x = d_s,
                             probs = 0.05,
                             na.rm = TRUE)]

  ## calculate water level mean
  h_mean[i] <- mean(h_ok,
                    na.rm = TRUE)

  ## calculate water level standard deviation
  h_sd[i] <- sd(h_ok,
                na.rm = TRUE)
}
```

Visualisation of the results

The output of the model exercise is visualised in two plots, one showing all modelled spectra and their corresponding water levels and another showing the independent water level time series, the precipitation record and the uncertainty-including seismic estimate of a water level.

```
## setup plot area
par(mfcol = c(2, 1),
    mar = c(4, 4, 3, 4))

## Plot A - modelled seismic spectra
plot(x = f,
     y = P[6,],
     ylim = c(-180, -95),
     col = NA,
     xlab = "f (Hz)",
     ylab = "Spectral power (dB)")

i_plot <- seq(from = 1,
              to = length(h_P),
              by = 10)

for(i in i_plot) {

  lines(x = f,
        y = P[i,],
        col = adjustcolor(rainbow(length(h_P))[i], alpha.f = 0.3))
}

## add three representative empiric spectra
lines(x = s_trunc[[2]]$f,
      y = 10 * log10(s_trunc[[2]]$s),
      lwd = 3,
      col = "darkblue")

lines(x = s_trunc[[6]]$f,
      y = 10 * log10(s_trunc[[6]]$s),
      lwd = 3,
      col = "black")

lines(x = s_trunc[[61]]$f,
      y = 10 * log10(s_trunc[[61]]$s),
      lwd = 3,
      col = "darkgreen")

## Plot B - time series of water level, precipitation and seismic data
plot(x = t,
     y = h_sd,
     col = NA,
     ylim = c(0, 1.5),
     xlab = "Time",
     ylab = "River stage (m)")

## uncertainty polygon of seismic model results
polygon(x = c(t,
              rev(t)),
        y = c(h_mean + h_sd,
              rev(h_mean - h_sd)),
        border = NA,
```

```
        col = "grey")

## average line of seismic model results
lines(x = t,
      y = h_mean,
      lwd = 2)

## water level
lines(w$Zeit,
      w$WBG_WB_PEGEL / 1000,
      lwd = 2,
      col = "blue")

## representative spectra timestamps visualisation
lines(x = c(t[2], t[2] + 3600),
      y = c(0, 0),
      lwd = 3,
      col = "darkblue")

lines(x = c(t[6], t[6] + 3600),
      y = c(0, 0),
      lwd = 3,
      col = "black")

lines(x = c(t[61], t[61] + 3600),
      y = c(0, 0),
      lwd = 3,
      col = "darkgreen")

## add precipitation data with own y axis
par(new = TRUE)

plot(x = clim$MESS_DATUM,
     y = -clim$R1,
     ylim = c(-8, 0),
     type = "h",
     col = "darkblue",
     axes = FALSE,
     ann = FALSE)

axis(side = 4)
mtext(side = 4,
      line = 3,
      text = "Precipitation (mm/h)",
      col = "darkblue")
```



References

Boessenkool, B., 2017. rdwd: Select and Download Climate Data from 'DWD' (German Weather Service), https://CRAN.R-project.org/package=rdwd, r package version 0.9.0.

Dietze, M., 2017. The R package 'eseis' – a comprehensive toolbox for environmental seismology. Submitted to Earth Surface Dynamics. [December 2017].

Gimbert, F., Tsai, V. C., and Lamb, M. P., 2014. A physical model for seismic noise generation by turbulent flow in rivers, J. Geophys. Res., 119, 2209–2238, https://doi.org/10.1002/2014JF003201.