Akustik 1

Praktikumsbericht

Christoph Zimmermann

Inhaltsverzeichnis

1. Praktikum: Zuglärm

Aufgabenstellung

http://www.isiweb.ee.ethz.ch/teaching/courses/ak1/pa1-eisenbahnE.htm

Aus der gegebenen Zuglärmmessung soll jeweils für die Tages- (6 – 22 Uhr) und Nachtzeit (22 – 6 Uhr) der Beurteilungspegel (Leq) ermittelt werden.

Es wird angenommen, dass während dem Tag 200 und in der Nacht 34 Züge vorbeifahren.

Vorgehen

Die Auswertung erfolgte mit Hilfe eines Python Scripts (siehe Anhang).

Als erstes wurde die gegebene Messung graphisch dargestellt um dann von Hand die Zeitpunkte der Zugvorbeifahrten je Richtung zu bestimmen. Es könnte auch versucht werden diese Ereignisse automatisiert zu bestimmen z. B. über einen Schwellwert. Durch die teilweise hohen Pegel der anderen Lärmquellen (Flugzeuge) ist dies aber mit einer relativ grossen Unsicherheit verbunden.

Aus diesen Ereignissen wurde jeweils der Lärmexpositionspegel (Le) ermittelt und daraus der mittlere Lärmexpositionpegel (Le_{avg}) je Richtung ermittelt.

Da in dieser Messung nur der Lärm der Züge von Interesse ist, muss für die weiteren Berechnungen der mittlere Hintergrundlärmpegel bestimmt werden. Dazu werden in der gegebenen Messung per Zufall Bereiche mit der gleichen Länge wie eine Zugdurchfahrt ausgewählt und daraus wieder der Lärmexpositionspegel (Le) berechnet. Damit eine statistisch signifikante Aussage getroffen werden kann, werden zehnmal mehr Hintergrundlärmbereiche als Zugdurchfahrten ausgewählt. Zur weiteren Berechnung wird nun der Wert ausgewählt der höher ist als 90% aller Hintergrundlärmpegel.

Mit den mittleren Lärmexpositionpegeln und dem ermittelten Hintergrundlärmpegel wurde dann der Beurteilungspegel (Leq) und dessen Unsicherheitsbereich für die Tages- und Nachtzeit bestimmt.

Resultate

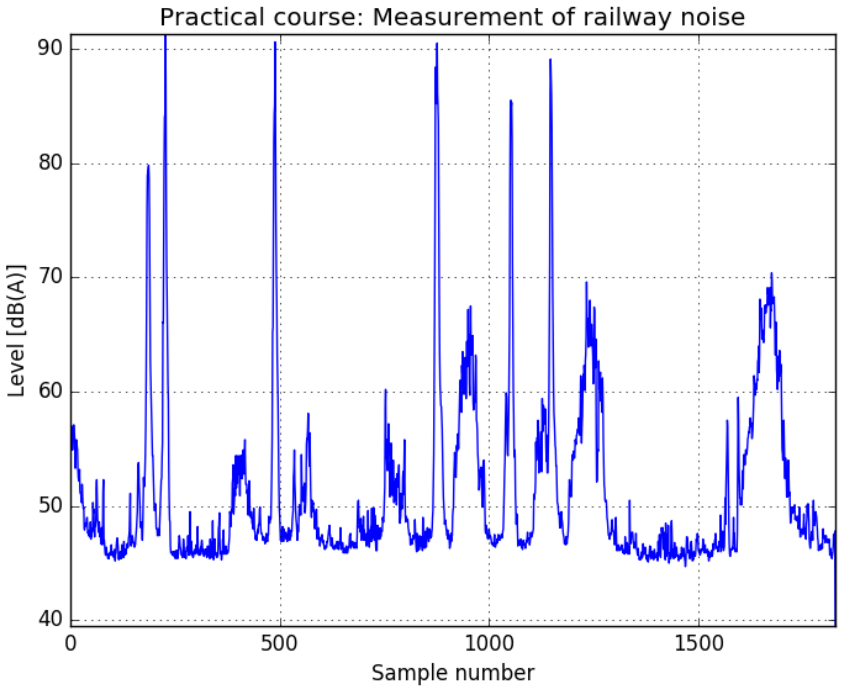


Abbildung 1: Verlauf des Lärmpegels während der Messdauer. Die hohen, schmalen Spitzen sind die Zugvorbeifahrten

Average sound exposure level by trains from Dübendorf: 93.761 dB

Average sound exposure level by trains from Stettbach: 93.287 dB

Sound exposure level caused by background noise (with 90% probability): 68.186 dB

Sound exposure level caused by trains from Dübendorf: 93.749 dB + 0.012 dB

Sound exposure level caused by trains from Stettbach: 93.273 dB + 0.013 dB

Equivalent continuous sound pressure level (Leq) during day: 68.923 dB + 0.013 dB

Equivalent continuous sound pressure level (Leq) during night: 64.238 dB + 0.013 dB

2. Praktikum: Absorptionsmessung mit dem kundtschen Rohr

Aufgabenstellung

http://www.isiweb.ee.ethz.ch/teaching/courses/ak1/pa1-kundtE.htm

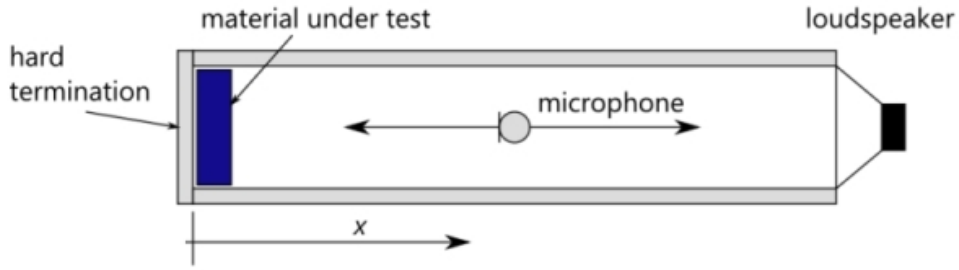


Abbildung 2: Aufbau eines kundtschen Rohrs

Durch Messungen mit dem kundtschen Rohr soll der Absorptionskoeffizient α ermittelt werden eines porösen Absorbers. Das untersuchte Material ist ein 3 cm dicker Schaumstoff der entweder direkt oder mit 10 cm Abstand vor dem Abschluss des Rohrs montiert wird.

Aus der Information wo die Maxima und Minima zu liegen kommen, soll die Phasenverschiebung berechnet werden die das Material verursacht.

Des weiteren soll im offenen kundtschen Rohr das Schallfeld untersucht werden und das dabei auftretende Phänomen erklärt werden.

Vorgehen

Die Auswertung erfolgte mit Hilfe einer OpenOffice Tabelle (siehe Anhang).

Die Absorptionskoeffizienten wurden gemäss Aufgabenstellung bzw. Skript berechnet.

Aus der Messung mit dem akustisch hartem Abschluss konnte die jeweilige Wellenlänge berechnet werden:

$$\lambda = | 4 \cdot (x_{min} - x_{max}) |$$

Die Phasenverschiebungen wurden bestimmt über die örtliche Verschiebung des ersten Minima (engl. „Node“) vor dem zu testenden Material.Über die Wellenlänge kann diese Distanz in einen Winkel umgerechnet werden. Dabei wurde angenommen: Erstens, dass die Reflexion an der Front des Schaumstoffs dominiert, entsprechend wurde die Referenzebene der Phase jeweils um die Ausdehnung des Absorbers verschoben. Zweitens, dass Phasenverschiebungen in Richtung Lautsprecher positiv und in Richtung Schaumstoff negativ sind.

Resultate

	Schaum (3 cm) direkt vor Abschluss			
Frequenz [Hz]	n	α	Δφ [rad]	Δφ [°]
125	18.949813831	0.1904527485	-0.108569735	-6.22058759
250	17.101792571	0.2087653692	-0.14321966	-8.20588204
500	13.173514424	0.262304984	-0.15707963	-8.999999874
1000	7.5836721712	0.4117117835	-0.168629604	-9.661764616
2000	2.6143556442	0.8005028641	0.0092399785	0.5294117693

	Schaum (3 cm) mit Abstand (10 cm) vor Abschluss			
Frequenz [Hz]	n	α	Δφ [rad]	Δφ [°]
125	12.468285038	0.2749426507	-0.351119171	-20.1176466
250	7.7058729125	0.4066840945	-0.399629062	-22.8970586
500	2.8252794396	0.7723160636	0.3349492231	19.191176836
1000	4.8283449801	0.5685488484	-0.025409938	-1.455882226
2000	3.1805923025	0.7279345985	-0.027719934	-1.588235218

Für die Diskussion der offenen Röhre wurde der Betrag des Reflexionsfaktors berechnet:

	Offen (ohne Abschluss)	
Frequenz [Hz]	|R|	Δφ [rad]
125	0.8678719237	-1.570796327
250	0.9986335084	0.7253383229
500	0.9397478872	0.3372592165
1000	0.7390152079	0.1409096746
2000	0.3488738947	0.0600598597

Es ist zu sehen, dass bei tieferen Frequenzen bei einer offenen Röhre der Reflexionsfaktor ungefähr gleich eins ist. Dies zeigt, dass:

- Die Werte der Impedanz in der Röhre und im Freifeld weit auseinander liegen
- Erwartungsgemäss sich an der Öffnung der Röhre ein Minimum bildet

Bei höheren Frequenzen gilt dies nicht mehr.

3. Praktikum: Raumakustische Messungen

Aufgabenstellung

http://www.isiweb.ee.ethz.ch/teaching/courses/ak1/pa1-raumE.htm

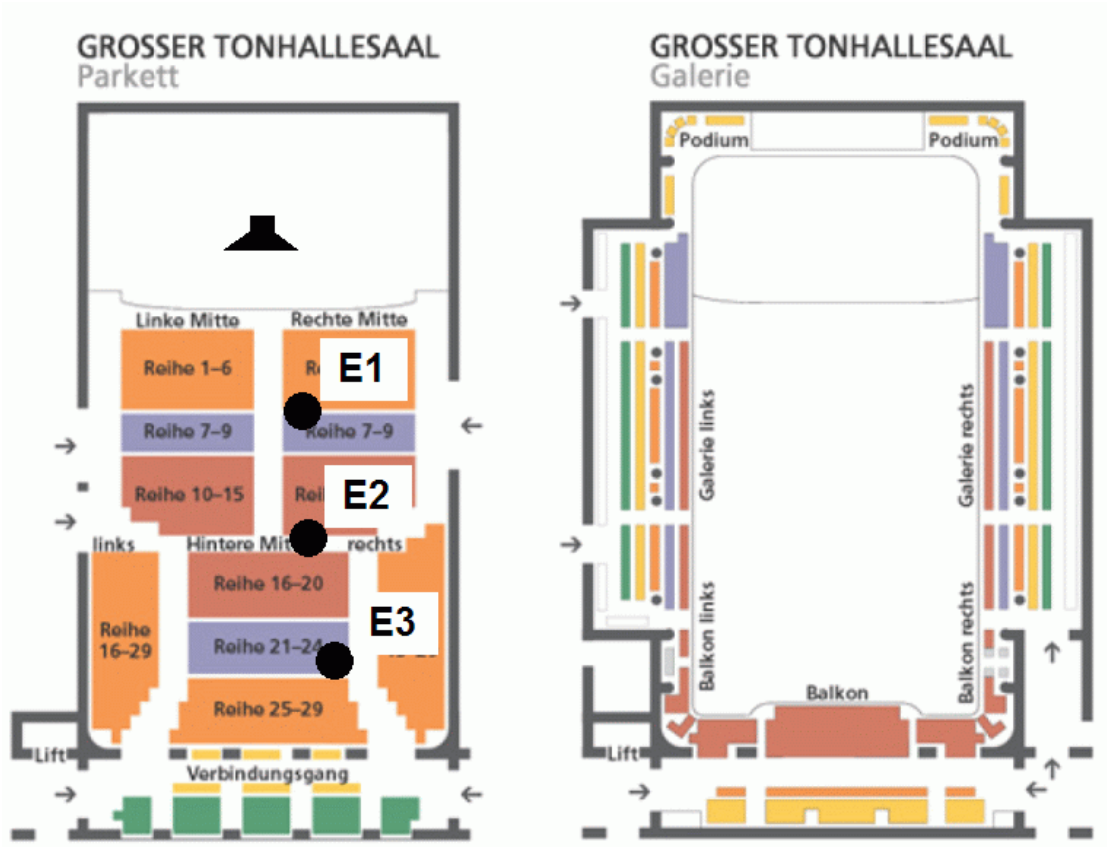


Abbildung 3: Position der drei Messungen in der Zürcher Tonhalle

Aus den Messungen der Impulsantworten an den drei Positionen E1, E2 und E3 sollen die statistischen Raumakustikkriterien RT, EDT und C80 bestimmt werden (jeweils für die Frequenz 500 und 2000 Hz)

Vorgehen

Die Auswertung erfolgte mit Hilfe eines Python Scripts (siehe Anhang).

Als erstes wurden die gegebenen Impulsantworten graphisch dargestellt, um dann von Hand die Zeitpunkte zu bestimmen wenn der Direktschall eintrifft. Es könnte auch versucht werden diese Ereignisse automatisiert zu bestimmen z. B. über einen Schwellwert. Weiter konnte in der graphischen Darstellung untersucht werden, ab welchem Zeitpunkt die Impulsantwort abgeklungen ist und der Hintergrundlärm dominiert.

Im zweiten Schritt wurde für die jeweilige Hörposition der Mittelwert des quadrierten Hintergrundlärms bestimmt und dieser Wert dann von der gesamten Impulsantwort abgezogen. Dies darf ge-

macht werden, weil die Impulsantwort mit einer Maximallängensequenz gemessen wurde (Das vorhandene Hintergrundrauschen wird über die gesamte Messsequenz gespreizt).

Aus den angepassten Impulsantworten wurden mit Hilfe der Schröder Inversintegration die Abfallkurven berechnet und daraus die Nachhallzeit (RT) und die Early-Decay-Time (EDT) berechnet. Zusätzlich werden die Abfallkurven graphisch dargestellt.

Als letztes wird aus den angepassten Impulsantworten die Klarheit C80 berechnet.

Resultate

Eintreffen des Direktschalls an Position E1: 0.031 s
Eintreffen des Direktschalls an Position E2: 0.050 s
Eintreffen des Direktschalls an Position E2: 0.072 s

Bereich wo die Impulsantwort abgeklungen ist und der Hintergrundlärm dominiert: 2.6 ... 3.0 s

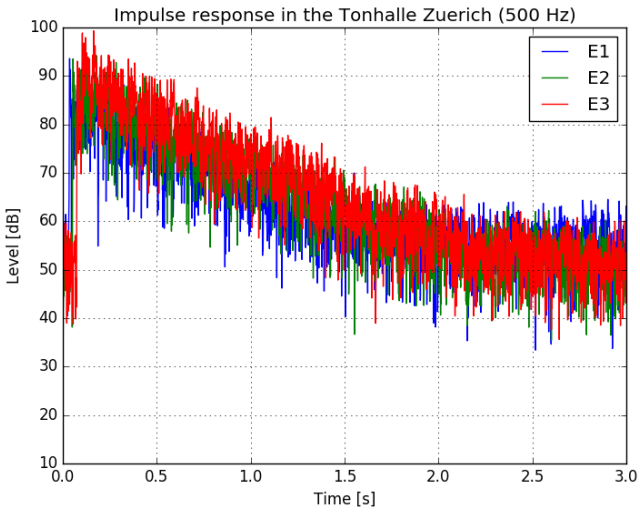


Abbildung 4: Impulsantworten bei 500 Hz

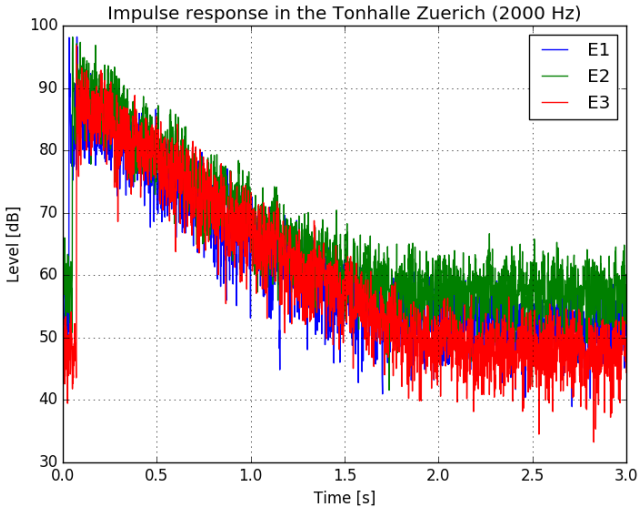


Abbildung 5: Impulsantworten bei 2000 Hz

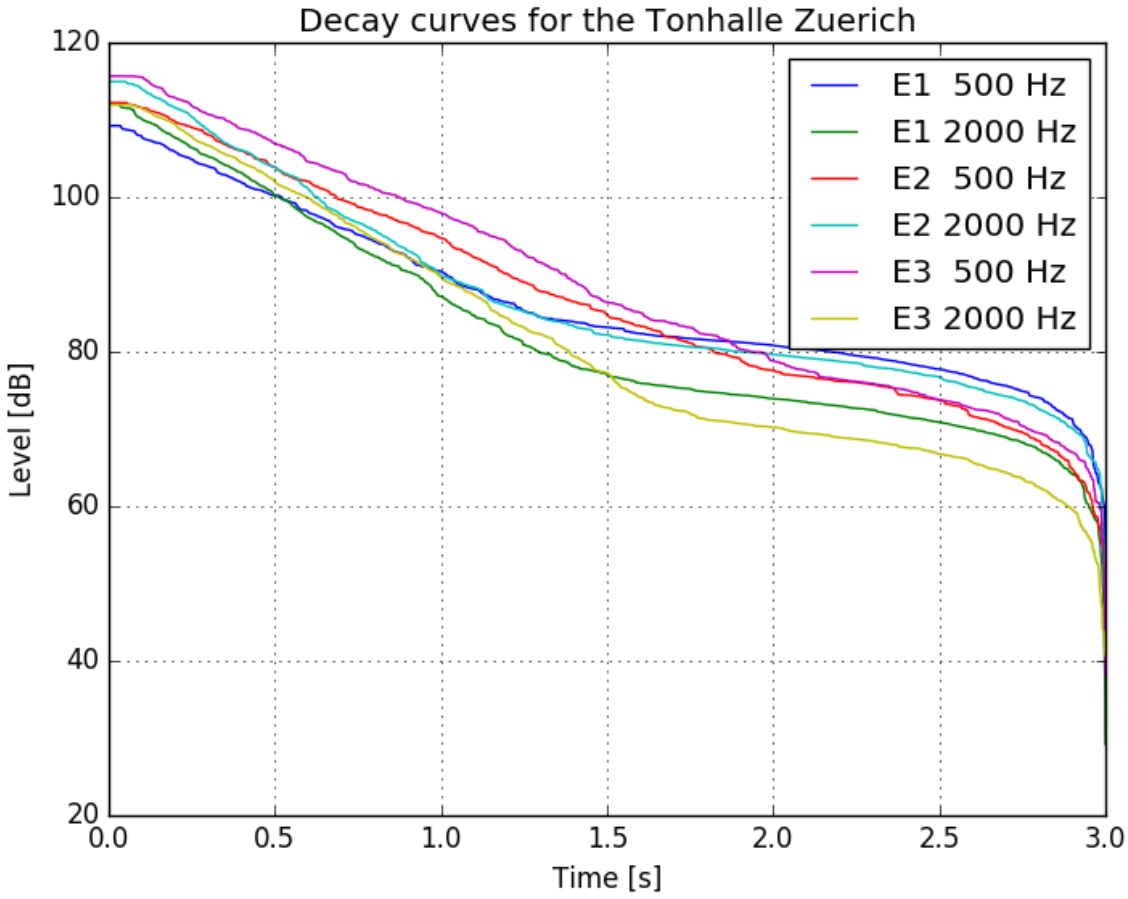


Abbildung 6: Abfallkurven aller Messpositionen

Statistische Kenngrössen der Hörpositionen:

	E1			**E2**			**E3**		
Frequenz [Hz]	RT [s]	EDT [s]	C80 [dB]	RT [s]	EDT [s]	C80 [dB]	RT [s]	EDT [s]	C80 [dB]
500	3.165	3.186	-2.945	3.036	3.144	-5.387	3.102	3.018	-3.974
2000	2.337	2.454	-1.840	2.208	2.382	-2.923	2.484	2.568	-3.541

Anhang

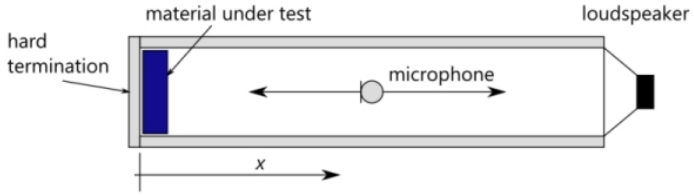
- Python Script „trainnoise.py2“
- OpenOffice Calc Dokument „Messungen Kundts Röhre.ods
- Python Script „pa1-raum.py2“

```
#!/usr/bin/env python2
# coding=utf-8
#
#Practical course: Measurement of railway noise
#
#Introduction
#
# The assessment of noise from railway lines is based on the yearly equivalent
# continuous sound pressure level (Leq), evaluated separately for day (6-22)
# and night (22-6). Railway noise consists of single events with pauses
# inbetween. It is therefore beneficial to calculate the Leq values from sound
# exposure levels (LE or SEL) that describe the energy of a train passage and
# taking into account the number of events.
#
# Sound exposure level LE or SEL und equivalent continuous sound pressure level
# Leq
#
# The sound exposure level LE or SEL is determined from the time dependent
# sound pressure p(t) as shown below. T corresponds to the integration or
# event time.
#
# The equivalent continuous sound pressure level Leq is determined from the
# time dependent sound pressure p(t) as shown below. T corresponds to the
# integration time.
#
# If a number of ni events with a sound exposure level LEi or SELi occur in the
# time period of interest T, the continuous equivalent sound pressure level Leq
# is given as:
#
#Measurement
#
# The measurement took place on Mai 26 2005 near Duebendorf at a distance of
# 10 m from the railway line. The microphone was installed 4 m above ground.
# During a period of 30 minutes, A-weighted short time Leq values were
# recorded every second. In parallel the microphone signal was recorded on DAT
# for future evaluation in the laboratory (Audio example of a train passage).
#
#Equipment in use:
#
#    Sound level meter: Norsonic 121, EMPA Nr. 1
#    DAT: HHB, EMPA Nr. B.

import math
import numpy as np
import matplotlib.pyplot as plt
import random

# During the measurement three train passages in direction Duebendorf
train_to_duebendorf = [
    #    09.28
    [177, 199],
    #    09.40
    [867, 890],
    #    09.44
    [1140, 1162]]

#and three passages in direction Stettbach
train_to_stettbach = [
    #    09.29
    [216, 235],
    #    09.33
    [481, 498],
    #    09.43
    [1037, 1063]]

# For the day and for the night period the equivalent continuous sound pressure
# level Leq of the ralway noise shall be determined. It is assumed that during
# the day (6-22) 200 train passages occur, during the night (22-6) 34 passages
# occur. It is assumed that the passages are equally distributed in both
# directions.

duration_day = (22-6)*3600
duration_night = 8*3600
train_passages_day = 200
train_passages_night = 34

# Sound pressure level reference level
p0 = 2e-5  # [pascal]

# Open the data file an import the sound pressure levels
```

```
I = []
with open('./pegelschriebE.txt', 'r') as tsv:
    for line in tsv:
        l = line.strip().split('\t')
        try:
            I.append(l[1])
        except(IndexError):
            print("Line could not be read")

train_noise = np.asarray(I).astype('float')
# create an index variable with the length of the available data
x = range(0, train_noise.size)

# The figure shows the measured A-weighted sound pressure level time history
# from 09.26 til 09.56.
plt.plot(x, train_noise)

plt.xlabel('Sample number')
plt.ylabel('Level [dB(A)]')
plt.title('Practical course: Measurement of railway noise'),
plt.grid(True)
plt.axis('tight')
plt.savefig("railway_noise.png")

# Calculate for each of the six train events the corresponding event level LE.
le_duebendorf = []
for train in train_to_duebendorf:
    p = [(10**(0.1*s)) for s in train_noise[train[0]:train[1]]]
    le_duebendorf.append(10*math.log10(sum(p)))

le_stettbach = []
for train in train_to_stettbach:
    p = [(10**(0.1*s)) for s in train_noise[train[0]:train[1]]]
    le_stettbach.append(10*math.log10(sum(p)))

# Calculate from the six LE values the average event level
# (For each direction seperate)
le_duebendorf_avg = 10*math.log10(np.mean([10**(0.1*s) for s in le_duebendorf]))
print("Average sound exposure level by trains from Dübendorf: {0:.3f} dB".format(le_duebendorf_avg))
le_stettbach_avg = 10*math.log10(np.mean([10**(0.1*s) for s in le_stettbach]))
print("Average sound exposure level by trains from Stettbach: {0:.3f} dB".format(le_stettbach_avg))


# With help of a statistical evaluation of the unwanted noise during the periods
# without train events, derive the uncertainty of the above results.
index_of_train_noise = train_to_duebendorf+train_to_stettbach
index_of_train_noise.sort()

# Generate an array with all the noise not comming from the trains
background_noise = []
background_noise.extend(train_noise[0:index_of_train_noise[0][0]])
for i in range(0, len(index_of_train_noise)-1):
    background_noise.extend(train_noise[index_of_train_noise[i][1]+1:index_of_train_noise[i+1][0]])

# Now we calculate sound exposure levels from the background noise
# For this we take random samples with the same lenght as the train events,
# to get a good guess we calculate ten times more events than actual train
# events have occoured.
number_of_background_events = 10*len(index_of_train_noise)
length_train_event = index_of_train_noise[0][1]-index_of_train_noise[0][0]
random.seed(train_noise[0])

le_background = []
for i in range(0, number_of_background_events):
    randint = random.randint(0, len(background_noise))
    p = [(10**(0.1*s)) for s in background_noise[randint:(randint+length_train_event)]]
    le_background.append(10*math.log10(sum(p)))

# Now from this exposure levels get the value which is higher than 90% of all
# background noise events (gives us a 90% probability of our calculated error)
le_background_90p = np.percentile(le_background, 90)
print("Sound exposure level caused by background noise (with 90% probability): {0:.3f} dB".format(le_background_90p))

le_duebendorf_train = 10*math.log10(10**(0.1*le_duebendorf_avg) -
                                    10**(0.1*le_background_90p))
print("Sound exposure level caused by trains from Dübendorf: {0:.3f} dB + {1:.3f} dB".format(le_duebendorf_train,
(le_duebendorf_avg-le_duebendorf_train)))
le_stettbach_train = 10*math.log10(10**(0.1*le_stettbach_avg) -
                                   10**(0.1*le_background_90p))
print("Sound exposure level caused by trains from Stettbach: {0:.3f} dB + {1:.3f} dB".format(le_stettbach_train,
```

```
    (le_stettbach_avg-le_stettbach_train)))
160
161
162 # Based on the average event levels for each direction. and with the number of
163 # events per day and night the Leq is calculated for the day and night period.
164 leq_day_with_noise = 10*math.log10((0.5*train_passages_day*10**(0.1*le_duebendorf_avg) +
165                                     0.5*train_passages_day*10**(0.1*le_stettbach_avg))
166                                    / duration_day)
167 leq_day = 10*math.log10((0.5*train_passages_day*10**(0.1*le_duebendorf_train) +
168                          0.5*train_passages_day*10**(0.1*le_stettbach_train))
169                         / duration_day)
170 print("Equivalent continuous sound pressure level (Leq) during day: {0:.3f} dB + {1:.3f} dB"
171     .format(leq_day, leq_day_with_noise-leq_day))
172
173 leq_night_with_noise = 10*math.log10((0.5*train_passages_night*10**(0.1*le_duebendorf_avg) +
174                                       0.5*train_passages_night*10**(0.1*le_stettbach_avg))
175                                      / duration_night)
176 leq_night = 10*math.log10((0.5*train_passages_night*10**(0.1*le_duebendorf_train) +
177                            0.5*train_passages_night*10**(0.1*le_stettbach_train))
178                           / duration_night)
179 print("Equivalent continuous sound pressure level (Leq) during night: {0:.3f} dB + {1:.3f} dB"
180     .format(leq_night, leq_night_with_noise-leq_night))
181
182 plt.show()
183
```

Practical course: Measurement of the absorption in Kundt's tube



Messungen

	Hart abgeschlossen				Schaum (3 cm) direkt vor Abschluss				Schaum (3 cm) mit Abstand (10 cm) vor Abschluss				Offen (ohne Abschluss)			
Frequenz [Hz]	max [V]	max [cm]	min [V]	min [cm]	max [V]	max [cm]	min [V]	min [cm]	max [V]	max [cm]	min [V]	min [cm]	max [V]	max [cm]	min [V]	min [cm]
125	1.990049834	0	0.009950169	68.00000004	1.889016006	3	0.099685201	66.30000053	1.747560786	13	0.140160478	65.80000038	1.999999344	64.90000092	0.141474403	0
250	1.990049834	68.00000035	0.009950169	34.00000001	1.88958077	64.80000043	0.110490217	30.80000025	1.77027934	63.70000156	0.229731188	29.70000017	1.999999538	31.40000026	0.001367426	65.40000086
500	1.893428413	34.00000002	0.025555876	17.00000001	1.781081716	30.20000017	0.135201713	13.2000001	1.433587757	27.50000015	0.507414501	44.5000003	1.874141991	14.60000021	0.058214274	31.60000024
1000	1.893142795	17.00000008	0.017510226	8.500000006	1.685341007	12.70000005	0.22223284	4.200000069	1.593511938	28.90000003	0.330032743	20.4000001	1.66728083	23.10000012	0.250219169	14.60000019
2000	1.89305128	8.500000026	0.037466781	4.200000008	1.382077105	11.90000002	0.528649233	7.600000012	1.459858524	20.20000008	0.458989517	16.00000006	1.290163064	19.60000007	0.622785313	6.800000014

Berechnungen

	Hart abgeschlossen				Schaum (3 cm) direkt vor Abschluss				Schaum (3 cm) mit Abstand (10 cm) vor Abschluss				Offen (ohne Abschluss)			
Frequenz [Hz]	n	α	λ [cm]		n	α	Δφ [rad]	Δφ [°]	n	α	Δφ [rad]	Δφ [°]	n	|R|	Δφ [rad]	Δφ [°]
125	200.0016127	0.019801332	272.0000002		18.94981383	0.190452748	-0.10856973	-6.22058759	12.46828504	0.274942651	-0.35111917	-20.1176466	14.13682827	0.867871924	-1.57079633	-90
250	200.0016127	0.019801332	136.0000013		17.10179257	0.208765369	-0.14321966	-8.20588204	7.705872912	0.406684095	-0.39962906	-22.8970586	1462.602161	0.998633508	0.725338323	41.55882462
500	74.08974745	0.052560177	68.00000004		13.17351442	0.262304984	-0.15707963	-8.99999987	2.82527944	0.772316064	0.334949223	19.19117684	32.19385674	0.939747887	0.337259216	19.32352971
1000	108.1164093	0.036322144	34.00000029		7.583672171	0.411711783	-0.1686296	-9.66176462	4.82834498	0.568548848	-0.02540994	-1.45588223	6.663281773	0.739015208	0.140909675	8.073529645
2000	50.52612498	0.076123898	17.20000007		2.614355644	0.800502864	0.009239978	0.529411769	3.180592303	0.727934598	-0.02771993	-1.58823522	2.07160162	0.348873895	0.06005986	3.441176476

```python
#!/usr/bin/env python2
# coding=utf-8
#
# Practical course: room acoustical measurement
#
# Introduction
#
#  The discussion of the acoustical quality of a room is based on the impulse
#  response. For given source and receiver position the impulse response
#  contains all possible information about the system - the impulse response
#  represents a finger print of the room for that particular source and
#  receiver position. From the impulse response, several objective criteria can
#  be evaluated such as
#
#  - reverberation time RT
#  - early decay time EDT
#  - clarity C80
#  - center time Ts
#  - strength G
#  - lateral energy fraction LF
#
#  The objective room acoustical criteria are frequency dependent.
#  The evaluation of the quantities is usually done in the octave bands from
#  125 Hz to 4 kHz.
#
#
# Impulse response measurements in the Tonhalle Zuerich
#  The impulse response measurements were performed in 2005. As a source an
#  omnidirectional loudspeaker was installed on the stage, with three different
#  receiver positions E1, E2 and E3.
#
#  The impulse response for point E2 can be heard in the file:
#  (be aware of the large dynamics of the signal)
#  http://www.isiweb.ee.ethz.ch/teaching/courses/ak1/e2.wav
#
#  The measured impulse responses were filtered in octave bands and squared.
#  For the evaluation these squared impulse responses are avaliable in the
#  500 Hz and 2 kHz band:
#
#    receiver E1
#    receiver E2
#    receiver E3
#
#
# Task
#
#  For the three receiver positions E1, E2 and E3, the room acoustical criteria
#  RT, EDT and C80 shall be evaluated in the two octave bands 500 Hz and 2 kHz.
#
#  RT and EDT are calculated by evaluating the Schroeder backward integration.
#  The RT measures the time for a decay of 60 dB. As such a dynamic is seldom
#  avaliable usually a decay of 30 dB is used (the corresponding time is then
#  multiplied by 2). However if the dynamics is not sufficient for 30 dB, the
#  decay range has to be lowered correspondingly. EDT evaluates the decay of
#  the top 10 dB (time is multiplied by 6).
#
#  C80 corresponds to the ratio of the integrals of the squared impulse
#  response, evaluated from 0 till 80 ms and from 80 ms till the end. It has to
#  be noticed that time 0 equals the moment of arrival of the direct sound.
#
#  As will be seen, the dynamics of the impulse response is rather low.
#  Therefore strategies have to be developed to handle the disturbing noise.
#  Of help is the fact, that the impulse responses were determined with a
#  correlation technique with maximum length sequences. From that follows that
#  any disturbing noise is equally distributed over the complete impulse
#  response.

import math
import numpy as np
import matplotlib.pyplot as plt

# File names of the recorded impulse responses at different locations
impulse_response_file_name = ['./e1.txt',  './e2.txt',  './e3.txt']

frequency = [' 500 Hz', '2000 Hz']
position = ['E1', 'E2', 'E3']

# Range of samples usable for background noise calculation (for example
# range before the direct sound arrives)
background_noise_samples = range(2600, 3000)
```

```
81
82  # Moment of arrival of the direct sound for each location (read out by hand)
83  impulse_responses_start = [
84      # for position E1, 500 Hz and 2000 Hz
85      31,
86      # for position E2, 500 Hz and 2000 Hz
87      50,
88      # for position E3, 500 Hz and 2000 Hz
89      72,
90  ]
91
92  # Sampling frequency
93  fs = 1000
94
95  # Sound pressure level reference level
96  p0 = 2e-5  # [pascal]
97
98  # check if a string can be represented as a number
99  def is_number(s):
100     try:
101         n=str(float(s))
102         if n == 'nan' or n=='inf' or n=='-inf' : return False
103     except ValueError:
104         try:
105             complex(s) # for complex
106         except ValueError:
107             return False
108     return True
109
110 # Open the data file an import the sound pressure levels
111 I = []
112 for file in impulse_response_file_name:
113     with open(file, 'r') as tsv:
114         time = []
115         octave_500_hz = []
116         octave_2000_hz = []
117         for line in tsv:
118             l = line.strip().split('\t')
119             if is_number(l[0]):
120                 try:
121                     time.append(l[0])
122                     octave_500_hz.append(l[1])
123                     octave_2000_hz.append(l[2])
124                 except(IndexError):
125                     print('Line could not be read')
126
127         I.append([time, octave_500_hz, octave_2000_hz])
128
129 impulse_responses = np.asarray(I).astype('float')
130
131 # The figure shows the measured sound pressure level time history (500 Hz)
132 pos = 0
133 for impulse_response in impulse_responses:
134     p = [10*math.log10(s) for s in impulse_response[1]]
135     plt.plot(impulse_responses[0][0], p, label='{0:s}'.format(position[pos]))
136     pos += 1
137
138 plt.xlabel('Time [s]')
139 plt.ylabel('Level [dB]')
140 plt.title('Impulse response in the Tonhalle Zuerich (500 Hz)')
141 plt.legend()
142 plt.grid(True)
143 plt.savefig('tonhalle_500hz.png')
144 plt.show()
145
146 # The figure shows the measured sound pressure level time history (2000 Hz)
147 pos = 0
148 for impulse_response in impulse_responses:
149     p = [10*math.log10(s) for s in impulse_response[2]]
150     plt.plot(impulse_responses[0][0], p, label='{0:s}'.format(position[pos]))
151     pos += 1
152 plt.xlabel('Time [s]')
153 plt.ylabel('Level [dB]')
154 plt.title('Impulse response in the Tonhalle Zuerich (2000 Hz)')
155 plt.legend()
156 plt.grid(True)
157 plt.savefig('tonhalle_2000hz.png')
158 plt.show()
159
160
```

```
# Calculate mean value of background noise at all positions
for impulse_response in impulse_responses:
    for freq in range(1, len(impulse_response)):
        background_noise = np.mean(impulse_response[freq][background_noise_samples])
        #print('Noise: {0:.3f}'.format(10*math.log10(background_noise)))
        background_noise = np.sqrt(background_noise)
        # Subtract the background noise from the measured impulse response
        impulse_response[freq] = (np.sqrt(impulse_response[freq]) - background_noise)**2


# Calculate Schröder inverse integration according to the presentation
decay_curves = []
pos = 0
for impulse_response in impulse_responses:
    decay_curve = []
    for freq in range(1, len(impulse_response)):
        last_index = impulse_response[freq].size-1
        decay_curve_lin = range(0, last_index+1)
        # The last sample of the decay curve equals the last sample of the
        # impulse_response because its the only sample to integrate over
        decay_curve_lin[last_index] = impulse_response[freq][last_index]
        # Calculate the integral
        for i in range(last_index-1, -1, -1):
            decay_curve_lin[i] = decay_curve_lin[i+1] + impulse_response[freq][i]

        p = [10*math.log10(s) for s in decay_curve_lin]
        decay_curve.append(p)
        plt.plot(impulse_responses[0][0], p,
                 label='{0:s} {1:s}'.format(position[pos], frequency[freq-1]))
    decay_curves.append(decay_curve)
    pos += 1

plt.xlabel('Time [s]')
plt.ylabel('Level [dB]')
plt.title('Decay curves for the Tonhalle Zuerich')
plt.legend()
plt.grid(True)
plt.savefig('tonhalle_decay_curves.png')
plt.show()

# For the reverberation time RT and the early decay time EDT the points where
# the sound pressure level decayed by 10 respectively 20 dB have to be found
# (RT is measured from 5 to 25 dB decay)
pos = 0
for decay_curve in decay_curves:
    print('RT and EDT for position {0:s}:'.format(position[pos]))
    for freq in range(0, len(decay_curve)):
        steady_level = decay_curve[freq][0]
        for i in range(len(decay_curve[freq])-1, 1, -1):
            if steady_level - decay_curve[freq][i] > 25:
                t25 = i
            if steady_level - decay_curve[freq][i] > 5:
                t5 = i
            if steady_level - decay_curve[freq][i] > 10:
                t10 = i
        rt = 3.0 * (t25-t5) / fs
        edt = 6.0 * (t10-impulse_responses_start[pos]) / fs
        print('  {0:s}: RT: {1:.3f} s EDT: {2:.3f} s'.format(
              frequency[freq], rt, edt))
    pos += 1

# After calculating T60 and EDT check if the condition is met to ignore the
# influence of the octave band filter response

# Calculate the clarity C80
c80 = []
for i in range(len(impulse_responses)):
    print('Clarity C80 for position {0:s}:'.format(position[i]))
    c80_temp = []
    start = impulse_responses_start[i]
    t80ms = start + 80e-3*fs
    for freq in range(1, len(impulse_responses[0])):
        clarity = 10*math.log10(
            sum(impulse_responses[i][freq][start:t80ms])/
            sum(impulse_responses[i][freq][t80ms:]))
        c80_temp.append(clarity)
        print('  {0:s}: {1:.3f} dB'.format(frequency[freq-1], clarity))
    c80.append(c80_temp)


```