ISS_Projekt

December 10, 2024

Jan Štefan Hodák - xhodakj00

[ ]:
```
# needed
import os
import re
import glob
import soundfile as sf
from IPython.display import Audio
from IPython.display import display
# recommended ...
from scipy import signal
from scipy.io import wavfile
from scipy.fft import fft, ifft, fftfreq
import scipy.io
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import cm
```

[ ]:
```
# read the file - change login to YOUR login
login = "xhodakj00"
zip_file = login + ".zip"
assignment_file = "https://www.fit.vut.cz/study/course/ISS/public/proj2024-25/
 ↪personal/" + zip_file
!wget $assignment_file
!unzip -o $zip_file
```

Pořadí testovacích i referenčních signálů se v průběhu řešení z mně neznámeho důvodu změnilo, proto prosím aby byl brán ohled o možné špatné pořadí vúči správnému řešení. Z tohoto důvodu jsem při finálním výpisu vypsal také i názvy jednotlivých nahrávek.

[ ]:
```
# load the data
# references will be in ref_signals, reference labels in ref_labels, reference␣
 ↪count in N_ref.
# tests  will be in test_signals, test labels in test_labels, test count in␣
 ↪N_test.
def get_signals(labs):
    signals = []
    N = len(labs)
```

```
    for car in labs:
      filename = login + "/" + car + ".wav"
      s, Fs = sf.read(filename)
      signals.append(s)
    return signals, N, Fs

def play_signals(signals, Fs):
  for signal in signals:
    display(Audio(signal, rate=Fs))

files = glob.glob(login + "/*.wav")
names = [re.sub(login + "/", "", s) for s in files]
labels = [re.sub(".wav", "",s) for s in names]
print ("----- test signals ---------")
r = re.compile("^test_");  test_labels = list(filter(r.match, labels))
print (test_labels);
test_signals, N_test, Fs = get_signals(test_labels); play_signals␣
 ↪(test_signals, Fs)
print ("----- reference signals ---------")
r = re.compile("(?!^test_)");  ref_labels = list(filter(r.match, labels))
print (ref_labels);
ref_signals, N_ref, Fs = get_signals(ref_labels); play_signals (ref_signals, Fs)
```

Jelikož jsou nekteré signály zašumněné, necháme je zpracovat filtrem, který je částečně očistí od šumu, aby se dalo se signály dále lépe pracovat

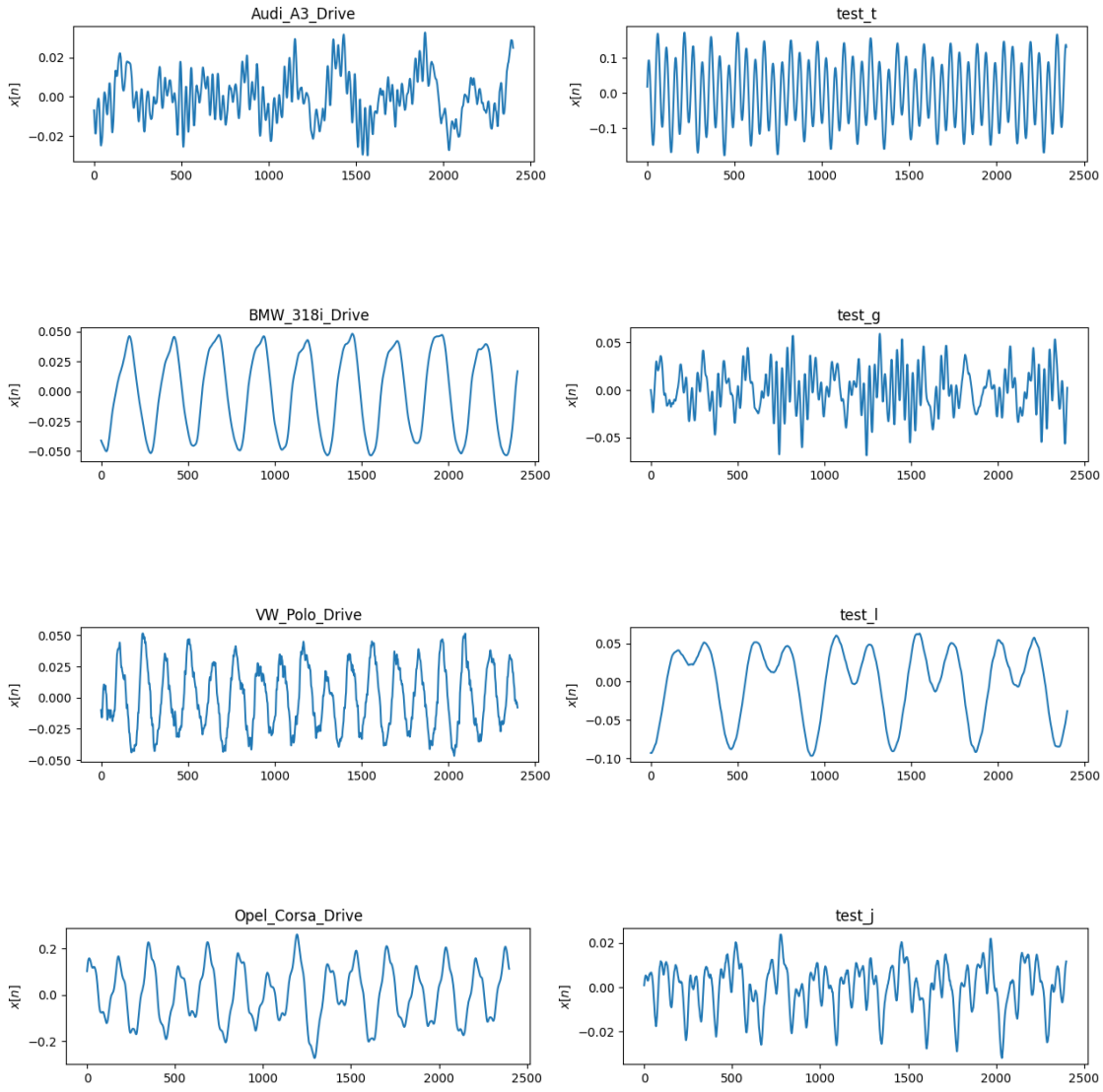
[ ]:
```
B = 0.1 * np.ones(10) # pole obsahující deset 0.1
for i in range(0,4):
  ref_signals[i] = signal.lfilter(B, 1, ref_signals[i])   # každý vzorek␣
 ↪necháme projít tímto lineárním filtrem
  test_signals[i] = signal.lfilter(B, 1, test_signals[i])
```

Nyní, když máme vcelku čisté signály můžeme začít s jejich analýzou. Nejprve ořežeme signály na kretší úsek, aby se s nimi dalo lépe přesněji pracovat a zároveň vezmeme pouze ty části signálu kde je motor vcelku na stabilních otáčkách něž zkazí zbytek nahrávky. Dále můžeme zkusit vykreslit krátký úsek jejich průběhu

[ ]:
```
start = int(0.05 * Fs) #vezmeme si pouze část signálu aby byl zřetelný průběh␣
 ↪jednotlivých signálů
end = int(0.2 * Fs)
for i in range(0,4):
    ref_signals[i] = ref_signals[i][start:end]
    test_signals[i] = test_signals[i][start:end]
```

[ ]:
```
for i in range(0,4):
  _, ax = plt.subplots(1,2, figsize=(15,2))         #vytvoření a popsání grafů
  ax[0].plot(ref_signals[i])
```

```
ax[0].set_title(str(ref_labels[i]))
ax[0].set_ylabel('$x[n]$')

ax[1].plot(test_signals[i])
ax[1].set_title(str(test_labels[i]))
ax[1].set_ylabel('$x[n]$')
```



Z průběhu signálu jde vidět že signál test_t je podezdřele pravidelný, tudíž by nemusel pocházet od auta. Dále můžeme vidět nějaké podobnosti mezi signály BMW-318i a test_l a také mezi trochu mezi Opel_Corsa a test_j.

[ ]:
```
test_not_suitable = -1 #pokud najdeme testovací signál, který nesedí k žádnému
↪autu uložíme jeho číslo do této proměnné
```

Než začneme s porovnáváním signálů si musíme vytvoříme nějaké funkce, kterými lze signály automaticky porovnávat.

[ ]:
```
def cross_correlation(ref_spectrum, test_spectrum):
    correlation = np.correlate(ref_spectrum, test_spectrum, mode='full')  # ␣
 ↪Křížová korelace
    max_corr = np.max(correlation)
    return max_corr

def match_signals(ref_spectrums, test_spectrums):
    matches = {}
    best_matches = {}
    for test_idx, test_signal in enumerate(test_spectrums):
        best_match = None
        best_corr = -np.inf
        for ref_idx, ref_signal in enumerate(ref_spectrums):  #korelujeme␣
 ↪spektrum každého testovacího signálu se všemi spektry referenčních signálů
            correlation = cross_correlation(ref_signal, test_signal)
            if correlation > best_corr:  # Uložíme nejlepší shodu
                best_corr = correlation
                best_match = ref_idx
            matches[test_idx*4+ref_idx] = (ref_idx, correlation)
        best_matches[test_idx] = (best_match, best_corr)
    return matches, best_matches


def euclidean_distance(x, y):
    diff = np.array(x) - np.array(y)  # Rozdíly mezi odpovídajícími prvky
    squared_diff = diff ** 2          # Čtverce rozdílů
    sum_squared = np.sum(squared_diff)  # Součet čtverců
    distance = np.sqrt(sum_squared)     # Druhá odmocnina
    return distance

def match_euclid(ref_spectrums, test_spectrums):
    matches = {}
    best_matches = {}
    for test_idx, test_signal in enumerate(test_spectrums):
        best_match = None
        best_dist = np.inf
        for ref_idx, ref_signal in enumerate(ref_spectrums): # počítáme␣
 ↪euklidovu vzdálenost pro spektrum každého testovacího signálu se všemi␣
 ↪spektry referenčních signálů
            distance = euclidean_distance(ref_signal, test_signal)
            if distance < best_dist:  # ulož nejlepší shodu
                best_dist = distance
                best_match = ref_idx
            matches[test_idx*4+ref_idx] = (ref_idx, distance)
```

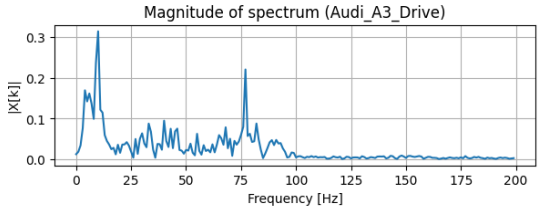
```
            best_matches[test_idx] = (best_match, best_dist)
    return matches, best_matches
```

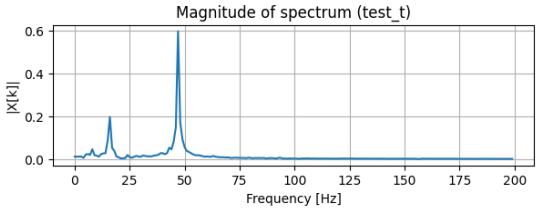
Nyní, když jsme analyzovali průbehy samotných signálů, můžeme začít s jejich dalším zpracováním. Vyzkoušíme provést samotnou diskrétní Furierovu transformaci pomocí FFT, čímž signál převedeme do frekvenční domény. Díky DFT můžeme analyzovat signál podle jeho chování na základě frekvence

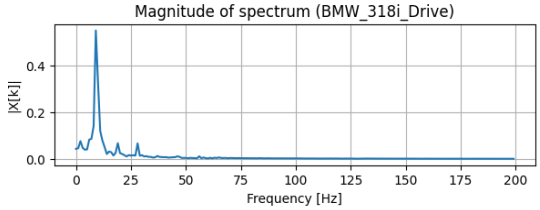
[ ]:
```
def normalize(spectrum):
    return spectrum / np.linalg.norm(spectrum)  #normalizace pomocí Euklidovské␣
 ↪normy

Ref_spectrums = [0,0,0,0]
Test_spectrums = [0,0,0,0] # pole pro uložení spekter jednotlivých signálů
N = 200;        #chci vytisknout jen prvních 600 vzorků
for i in range(0,4):
    Ref = np.fft.fft(ref_signals[i])
    Test = np.fft.fft(test_signals[i])  # provedu fft pro jednotlivé signály
    kall = np.arange(0,N)
    Ref_spectrums[i] = normalize(np.abs(Ref))
    Test_spectrums[i] = normalize(np.abs(Test))

    _, ax = plt.subplots(1,2, figsize=(15,2))   #vytvoření a popsání grafů
    ax[0].grid()
    ax[0].plot(kall, (Ref_spectrums[i][0:N]))
    ax[0].set_title("Magnitude of spectrum (" + str(ref_labels[i])+ ")")
    ax[0].set_xlabel('Frequency [Hz]')
    ax[0].set_ylabel('|X[k]|')

    ax[1].plot(kall,(Test_spectrums[i][0:N]))
    ax[1].grid()
    ax[1].set_title("Magnitude of spectrum (" + str(test_labels[i])+ ")")
    ax[1].set_xlabel('Frequency [Hz]')
    ax[1].set_ylabel('|X[k]|')

matches, best_matches = match_euclid(Ref_spectrums, Test_spectrums) #spočítáme␣
 ↪korelace pro jednotlivá spektra

# Výsledky - zakomentářované vzdálenosti pro všechny kombinace, kromě␣
 ↪nejlepších z důvodu přehlednosti
#for test_idx, (ref_idx, dist) in matches.items():  #vytisknutí spočítaných␣
 ↪korelací mezi jednotlivými signály
#    print(f"Test signal {(int)((test_idx-ref_idx)/4)} matches Reference signal␣
 ↪{ref_idx} with distance of {dist:.2f}.")
for test_idx, (ref_idx, dist) in best_matches.items(): #vytisknutí nejlepších␣
 ↪shod jednotlivých signálů
    print(f"Test signal {test_idx} matches best with Reference signal {ref_idx}␣
 ↪with distance of {dist:.2f}")
```

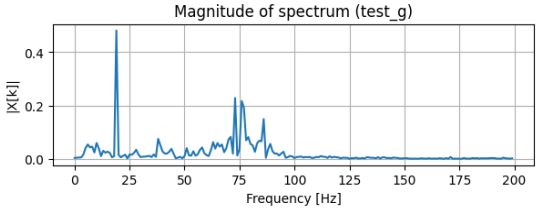
```
    if dist > 1.2: # pokud je signál příliš rozdílný od ostatních můžeme jej␣
↪vyřadit
        test_not_suitable = test_idx
```

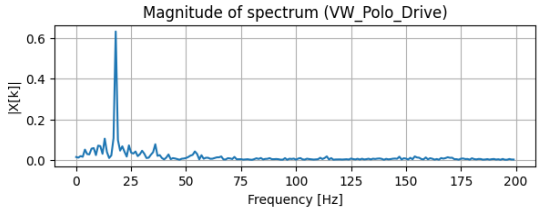
```
Test signal 0 matches best with Reference signal 0 with distance of 1.24
Test signal 1 matches best with Reference signal 0 with distance of 1.04
Test signal 2 matches best with Reference signal 0 with distance of 0.89
Test signal 3 matches best with Reference signal 0 with distance of 1.00
```

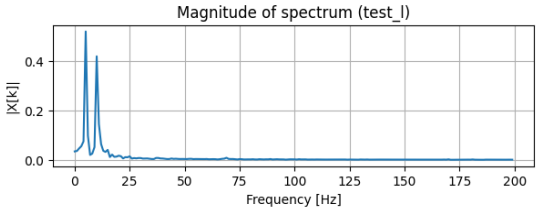


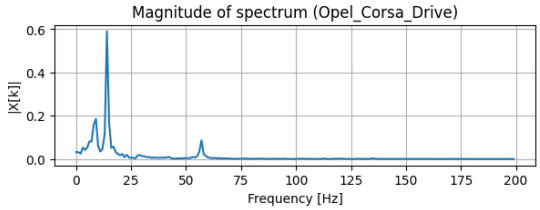


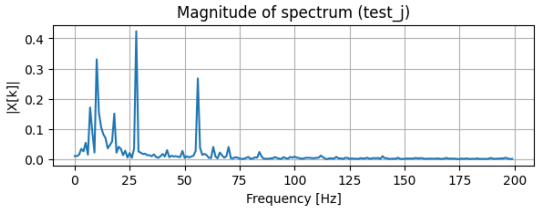












Z grafů velikosti spektra signálů jdou opět vyčíst nějaké podobnosti mezi signály. Například vysoký impuls na 20 Hz v spektrech vW_Polo a test_g. Pro každou kombinaci signálů si spočítáme ekli-

dovskou vzdálenost mezi nimi. Vypočítené hodnoty nám stále nestačí pro aspoň trochu spolehlivé vyhodnocení všech signálů. Můžeme ale naopak vyřadit test, který překročil stanovenou hranici pro vyřazení signálu, který nepatří žádnému z aut.

Další způsob jak analyzovat signál je převedení jeho spektra na PSD neboli spektrální hustotu výkonu, která udává energii signálu na dané frekvenci. Spektrální hustotu výkonu jsem nadále transformoval logaritmem aby více odpovídala lidskému slyšení.

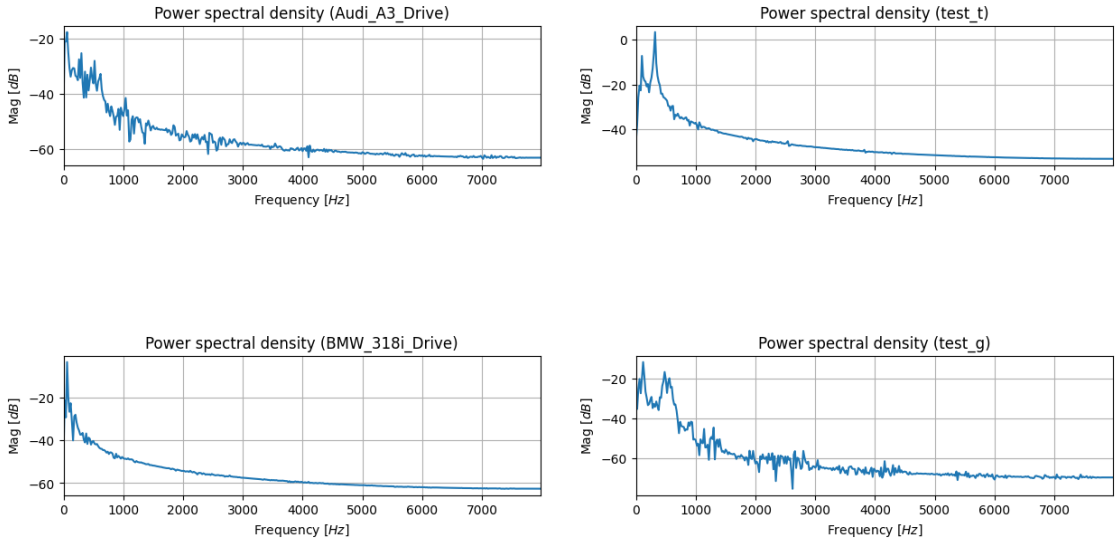
$$G[k] = 10 \log_{10} \frac{|X[k]|^2}{N}$$

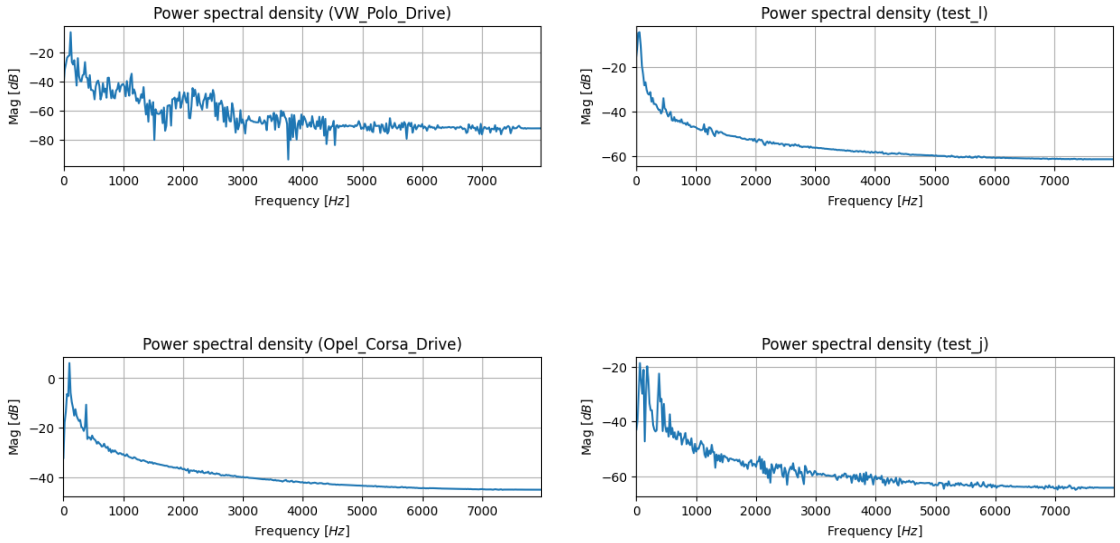
Kód pro výpočet PSD jsem založil na kódu z BP Petra Pálky

[ ]:
```
Ref_psd = [0,0,0,0]
Test_psd = [0,0,0,0]
for i in range(0,4):
    start = 0.10
    end = 0.15   #start and end of the signal segment in seconds
    Ref_seg = ref_signals[i][int(start * Fs):int(end * Fs)+1]   # vezmeme si␣
 ↪pouze určitý segment zvuku pro čistější výsledek
    Test_seg = test_signals[i][int(start * Fs):int(end * Fs)+1]
    N = Ref_seg.size
    Ref_spectrum = np.fft.fft(Ref_seg)
    Test_spectrum = np.fft.fft(Test_seg)   # spočítáme dft daného vzorku

    _, ax = plt.subplots(1,2, figsize=(15,2))

    Mag_ref = 10 * np.log10(1/N * np.abs(Ref_spectrum)**2) #ze spektra signálu␣
 ↪spočítáme jeho spektrální hustotu
    freq_ref = np.arange(Mag_ref.size) * Fs / N
    Ref_psd[i] = Mag_ref
    half = freq_ref.size//2

    ax[0].plot(freq_ref[:half], Mag_ref[:half])  # vykreslení grafu
    ax[0].set_xlim(freq_ref[:half].min(), freq_ref[:half].max())
    ax[0].set_title("Power spectral density (" +str(ref_labels[i])+ ")")
    ax[0].set_xlabel('Frequency $[Hz]$')
    ax[0].set_ylabel('Mag $[dB]$')
    ax[0].grid()


    Mag_test = 10 * np.log10(1/N * np.abs(Test_spectrum)**2) #ze spektra␣
 ↪signálu spočítáme jeho spektrální hustotu
    freq_test = np.arange(Mag_test.size) * Fs / N
    Test_psd[i] = Mag_test
    half = freq_test.size//2

    ax[1].plot(freq_test[:half], Mag_test[:half]) # vykreslení grafu
    ax[1].set_xlim(freq_test[:half].min(), freq_test[:half].max())
```

```
    ax[1].set_title("Power spectral density (" + str(test_labels[i])+ ")")
    ax[1].set_xlabel('Frequency $[Hz]$')
    ax[1].set_ylabel('Mag $[dB]$')
    ax[1].grid()

matches, best_matches = match_euclid(Ref_psd, Test_psd) # vypočítání␣
 ↪euklidových vzdáleností

# - zakomentářované vzdálenosti pro všechny kombinace, kromě nejlepších z␣
 ↪důvodu přehlednosti
#for test_idx, (ref_idx, dist) in matches.items():       # tisk
#    print(f"Test signal {(int)((test_idx-ref_idx)/4)} matches Reference signal␣
 ↪{ref_idx} with distance of {dist:.2f}.")
for test_idx, (ref_idx, dist) in best_matches.items():
    print(f"Test signal {test_idx} matches best with Reference signal {ref_idx}␣
 ↪with distance of {dist:.2f}.")
    if test_not_suitable == -1 and dist > 210.0:
        test_not_suitable = test_idx
```

```
Test signal 0 matches best with Reference signal 3 with distance of 225.62.
Test signal 1 matches best with Reference signal 2 with distance of 194.11.
Test signal 2 matches best with Reference signal 1 with distance of 68.61.
Test signal 3 matches best with Reference signal 1 with distance of 78.97.
```



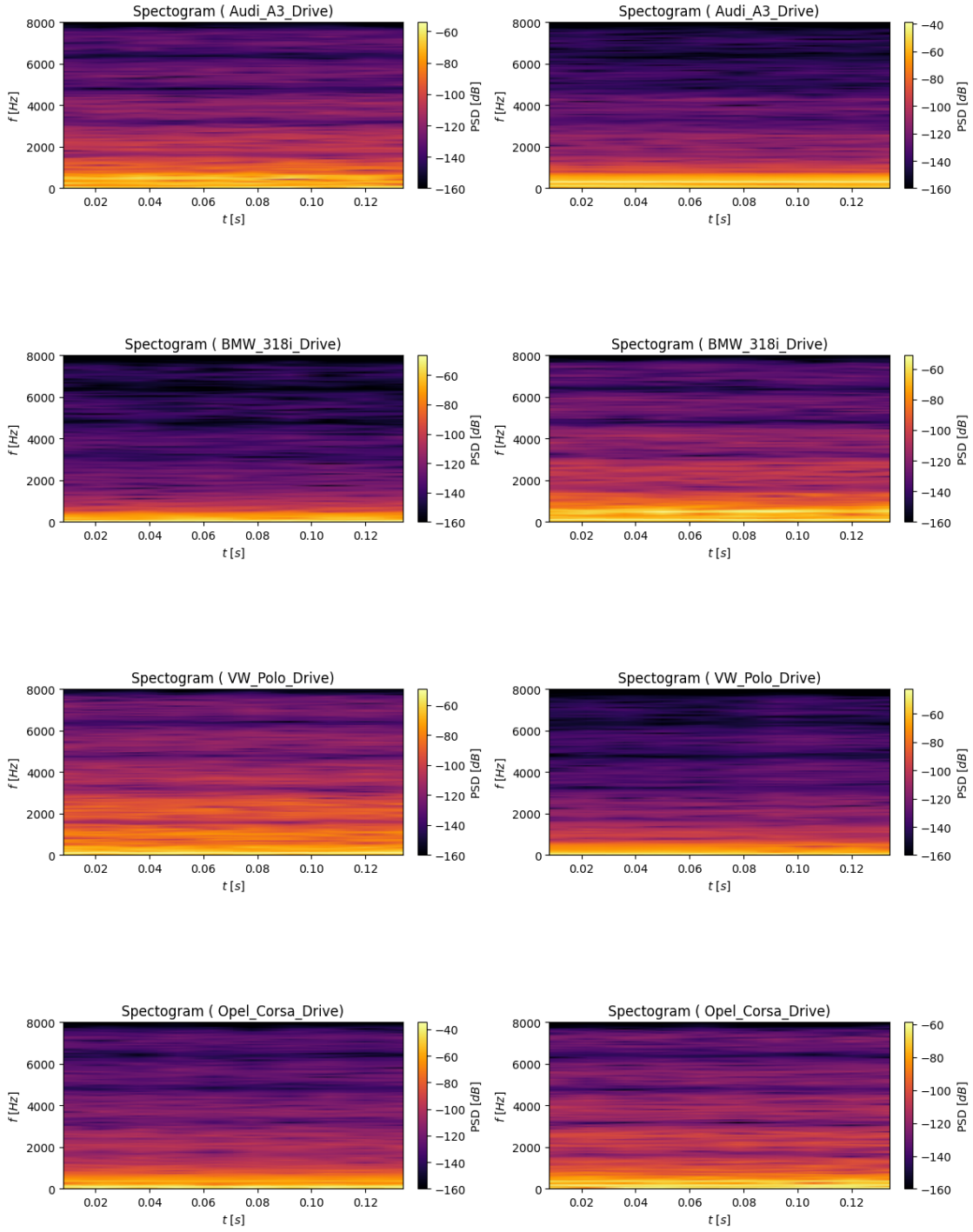


Na grafech spektrální hustoty výkonu, můžeme vidět další podobnosti, které si opět ověříme spočítáním euklidovských vzdáleností. Spočtené hodnoty, ale stále nestačí k jednoznačnému výsledku, ale i zde je možné vyřadit nehodící se nahrávku.

Další možnost, která mě napadla byla vytvoření spektrogramů, ta bohužel nevedla nikam dál, jelikož spektrogramy neukázaly žádné nové informace.

[ ]:
```
def plot_spectrogram(f, t, sgr, vmin=-160, ax=None):
    # Transfer to PSD
    sgr_log = 10 * np.log10(sgr + 1e-20)  # log(0) is undefined -> +1e-20 (add␣
 ↪small value)

    name = ''
    ax.set_title("Spectogram ( " +str(ref_labels[i])+ ")")
    ax.set_xlabel('$t\ [s]$')
    ax.set_ylabel('$f\ [Hz]$')
    ax.set_xlim(min(t), max(t))
      # pcolormesh of spectogram:
    cmesh = ax.pcolormesh(t, f, sgr_log, shading="gouraud", cmap=cm.inferno,␣
 ↪vmin=vmin)
    cbar = plt.colorbar(cmesh, ax=ax, fraction=0.046, pad=0.04)
    cbar.set_label('PSD $[dB]$')
    plt.tight_layout()

for i in range(0,4):
    _, ax = plt.subplots(1,2, figsize=(12,3))
    ref_freq, ref_time, ref_sgr = scipy.signal.spectrogram(ref_signals[i], Fs) ␣
 ↪  #výpočet spektrogramu
```

```
  test_freq, test_time, test_sgr = scipy.signal.spectrogram(test_signals[i],␣
↪Fs)
  plot_spectrogram(ref_freq, ref_time, ref_sgr, ax=ax[0])
  plot_spectrogram(test_freq, test_time, test_sgr, ax=ax[1])
```

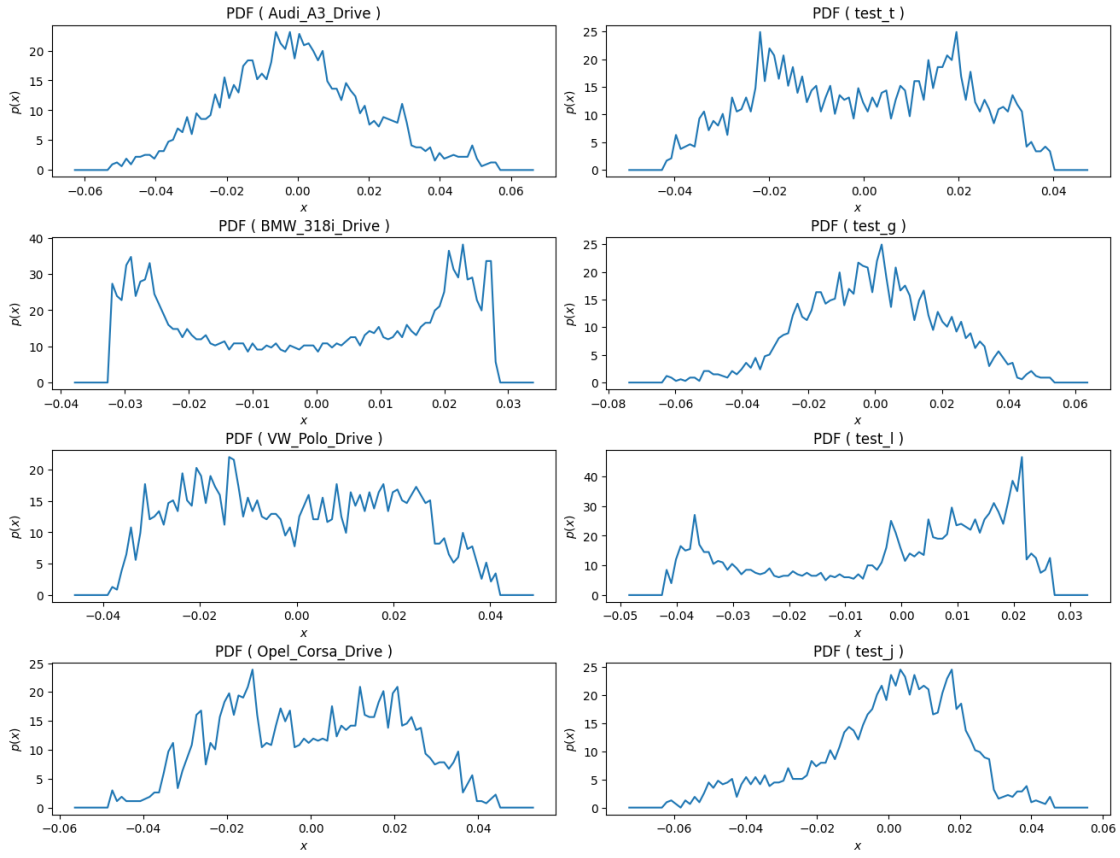


Na další způsob jak zpracovat signály jsem narazil při procházení jednotlivých prezentaci. A to je spočítat jejich funkci hustoty rozdělení pravděpodobnosti, kterou jsem našel v prezentaci *random_2*, ze které jsem následně převzal i kód pro její výpočet

[ ]:
```
_,ax = plt.subplots(4,2, figsize=(16,12))
plt.subplots_adjust(wspace=0.1, hspace=0.4)
Ref_spectrums = [0,0,0,0]
Test_spectrums = [0,0,0,0]

for i in range(0,4):
    for j in range(0,2):
        if j == 0:
            x = normalize(ref_signals[i])
            ax[i,j].set_title('PDF ( ' + str(ref_labels[i]) + " )")
        else:
            x = normalize(test_signals[i])
            ax[i,j].set_title('PDF ( ' + str(test_labels[i]) + " )")
        ###
        N = x.size
        datamin = np.min(x)
        datamax = np.max(x)
        datarange = datamax - datamin
 ↪                          #Převzatý úsek kódu z notebooku random_2
        bins = np.linspace(datamin-0.1*datarange,datamax+0.1*datarange,100)   #␣
 ↪let the data decide ...
        hist,_ = np.histogram(x,bins=bins)
        DELTA = bins[1] - bins[0]
        binsvisu = bins[:-1]
        px = hist / N / DELTA
        ###
        if j == 0:
            Ref_spectrums[i] = px
        else:
            Test_spectrums[i] = px

        ax[i,j].plot(binsvisu,px)
        ax[i,j].set_xlabel('$x$')
        ax[i,j].set_ylabel('$p(x)$')

matches, best_matches = match_euclid(Ref_spectrums, Test_spectrums) #spočítáme␣
 ↪vzdálenosti pro jednotlivá spektra
# Výsledky - zakomentářované vzdálenosti pro všechny kombinace, kromě␣
 ↪nejlepších z důvodu přehlednosti
#for test_idx, (ref_idx, corr) in matches.items():  #vytisknutí spočítaných␣
 ↪korelací mezi jednotlivými signály
#    print(f"Test signal {(int)((test_idx-ref_idx)/4)} and Reference signal␣
 ↪{ref_idx} matches with correlation of {corr:.2f}.")
```

```
for test_idx, (ref_idx, corr) in best_matches.items(): #vytisknutí nejlepších␣
↪shod jednotlivých signálů
    print(f"Test signal {test_idx} matches Reference signal {ref_idx} with␣
↪correlation of {corr:.2f}.")
```

```
Test signal 0 matches Reference signal 2 with correlation of 37.80.
Test signal 1 matches Reference signal 0 with correlation of 36.54.
Test signal 2 matches Reference signal 1 with correlation of 82.90.
Test signal 3 matches Reference signal 3 with correlation of 58.38.
```



Po vytvoření grafů funkcí hustoty rozdělení pravděpodobnosti, můžeme vidět hodně společných prvků jednotlivých grafů. Proto opět spočítáme euklidovské vzdálenosti. Tentokrát nám naše vypočtené hodnoty jsou schopny jednoznačně určit který signál patří jakému, pokud jsme už dříve určili nehodící se testovací signál.

Nyní si můžeme nechat vytisknout jak dopadly výsledky analýzy.

[ ]:
```
for test_idx, (ref_idx, corr) in best_matches.items(): #vytisknutí nejlepších␣
↪shod jednotlivých signálů
    if test_idx == test_not_suitable:
```

```
        print(f"\033[31mTest signal {test_idx} ({test_labels[test_idx]}) does␣
↪not match any of the reference signals.")
    else:
        print(f"\033[32mTest signal {test_idx} ({test_labels[test_idx]})␣
↪belongs to Reference signal {ref_idx} ({ref_labels[ref_idx]})")
```

```
Test signal 0 (test_t) does not match any of the reference signals.
Test signal 1 (test_g) belongs to Reference signal 0 (Audi_A3_Drive)
Test signal 2 (test_l) belongs to Reference signal 1 (BMW_318i_Drive)
Test signal 3 (test_j) belongs to Reference signal 3 (Opel_Corsa_Drive)
```