ISS 2024 protokol

- autor: Juraj Mesík
- login: xmesikj00

```
import os
import re
import glob
import soundfile as sf
from IPython.display import Audio
from IPython.display import display


from scipy.fft import fft, fftfreq
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile


login = "xmesikj00"
zip_file = login + ".zip"
assignment_file = "https://www.fit.vut.cz/study/course/ISS/public/proj2024-25/p
!wget $assignment_file
!unzip -o $zip_file
```

```
--2024-12-16 19:30:56--  https://www.fit.vut.cz/study/course/ISS/public/pro
Resolving www.fit.vut.cz (www.fit.vut.cz)... 147.229.9.65, 2001:67c:1220:80
Connecting to www.fit.vut.cz (www.fit.vut.cz)|147.229.9.65|:443... connecte
HTTP request sent, awaiting response... 200 OK
Length: 217662 (213K) [application/zip]
Saving to: ‘xmesikj00.zip’

xmesikj00.zip       100%[===================>] 212.56K   232KB/s    in 0.9s

2024-12-16 19:30:57 (232 KB/s) - ‘xmesikj00.zip’ saved [217662/217662]

Archive:  xmesikj00.zip
   creating: xmesikj00/
  inflating: xmesikj00/BMW_1_Drive.wav
  inflating: xmesikj00/test_r.wav
  inflating: xmesikj00/Opel_Corsa_Drive.wav
  inflating: xmesikj00/test_m.wav
  inflating: xmesikj00/Subaru_Forester_Drive.wav
  inflating: xmesikj00/test_o.wav
  inflating: xmesikj00/BMW_318i_Drive.wav
  inflating: xmesikj00/test_c.wav
```

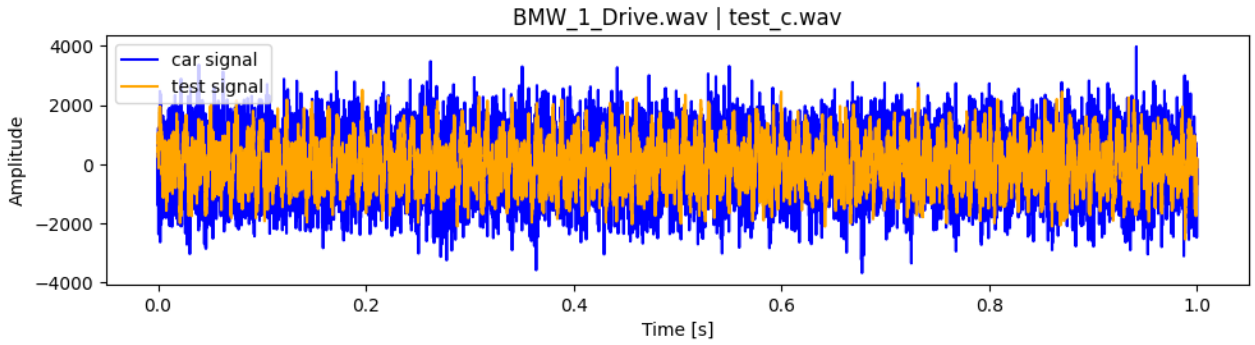
1 - Zobrazenie priebehu a analýza všetkých možných dvojíc signálov na jednom grafe

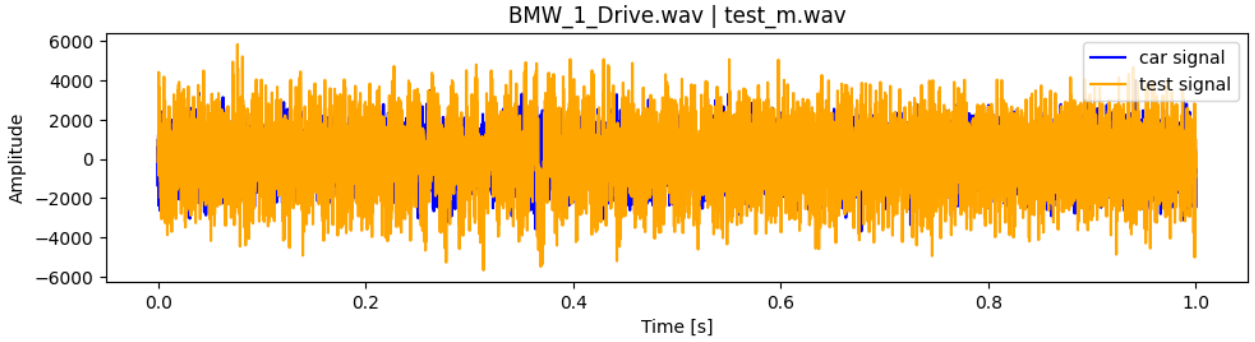
Na začiatok chcem jednoducho zistiť ako vyzerajú dvojice nahrávok na grafe. Pre presnejšie porovnanie sa pokúsim zobraziť jednotlivé neznáme signály tak, aby sa amplitúdou podobali na referenčné.

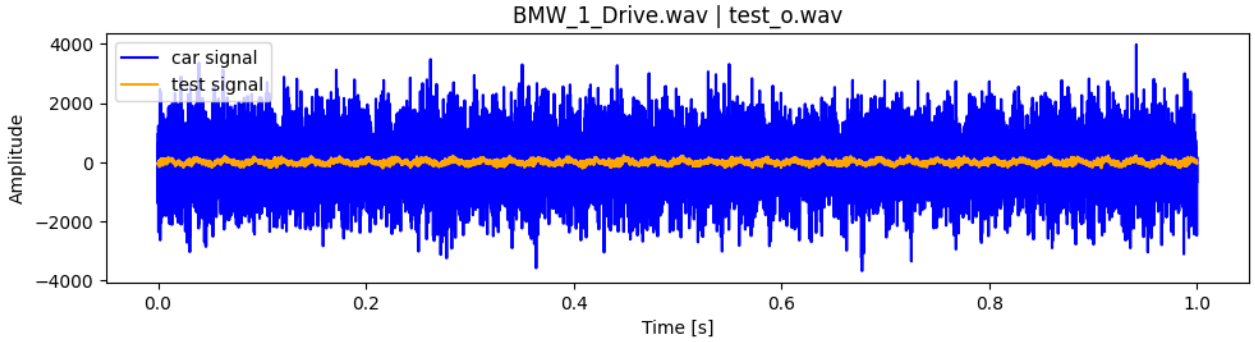
Na dosiahnutie podobnosti amplitúd som potreboval získať hodnotu, ktorou by som mohol jeden z porovnávaných signálov vynásobiť. Najprv som získal priemernú hodnotu oboch porovnávaných signálov cez funkciu 'np.mean()', následne do premennej 'scaling_factor' uložil podiel priemeru referenčnej nahrávky a neznámej nahrávky. Týmto získam hodnotu, ktorou keď vynásobím neznámy signál, signály by sa mali podobať amplitúdou.

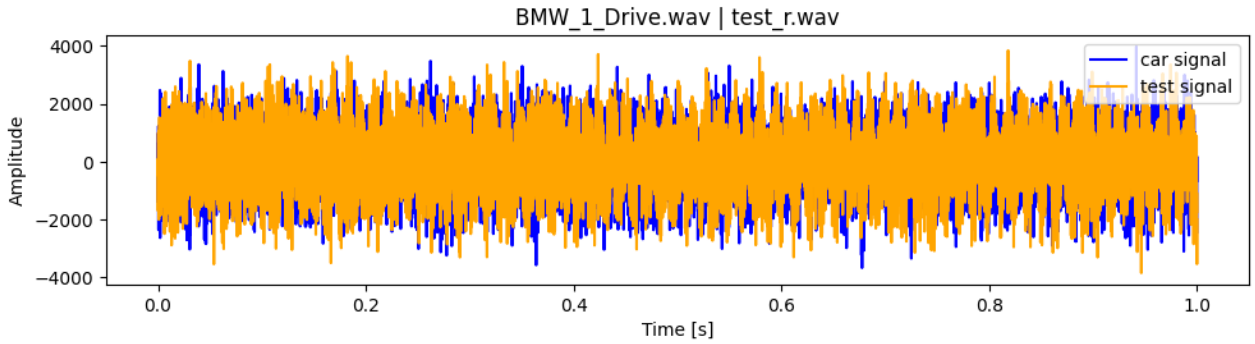
```
car_files = ['/content/xmesikj00/BMW_1_Drive.wav', '/content/xmesikj00/BMW_318i
test_files = ['/content/xmesikj00/test_c.wav', '/content/xmesikj00/test_m.wav',

test = []

#cyklus aby bol zobrazeny kazdy zvuk auta bude zobrazeny s kazdym neznamym zvuk
for car in car_files:

  samplerate, car_sound = wavfile.read(car)
  time = np.linspace(0, len(car_sound) / samplerate, len(car_sound))

  for test in test_files:

    samplerate, test_sound = wavfile.read(test)

    scaling_factor = abs(np.mean(car_sound)) / abs(np.mean(test_sound))

    # upravim neznamy signal
    test_sound_scaled = test_sound * scaling_factor
    #print(np.correlate(car_sound, test_sound_scaled))
    #test.append(np.correlate(car_sound, test_sound_scaled))

    # zobrazenie dvojice nahravok
    plt.figure(figsize=(10, 5))
    plt.subplot(2, 1, 2)
    plt.plot(time, car_sound, label="car signal", color='blue')
    plt.title(car[19:]+' | '+test[19:])
    plt.xlabel("Time [s]")
    plt.ylabel("Amplitude")
    plt.grid()
    plt.legend()

    plt.subplot(2, 1, 2)
    plt.plot(time, test_sound_scaled , label="test signal", color='orange')
    plt.xlabel("Time [s]")
    plt.ylabel("Amplitude")
    plt.grid()
    plt.legend()
```

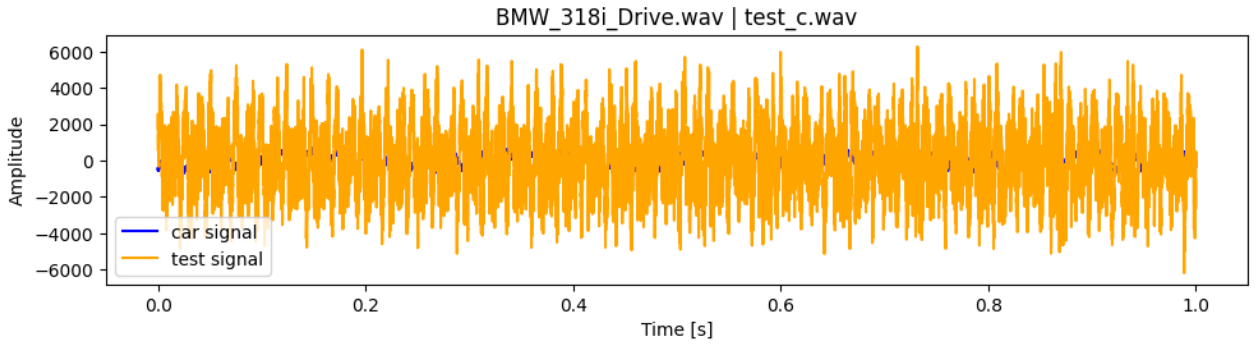
```
plt.legend()

plt.tight_layout()
plt.show()
```



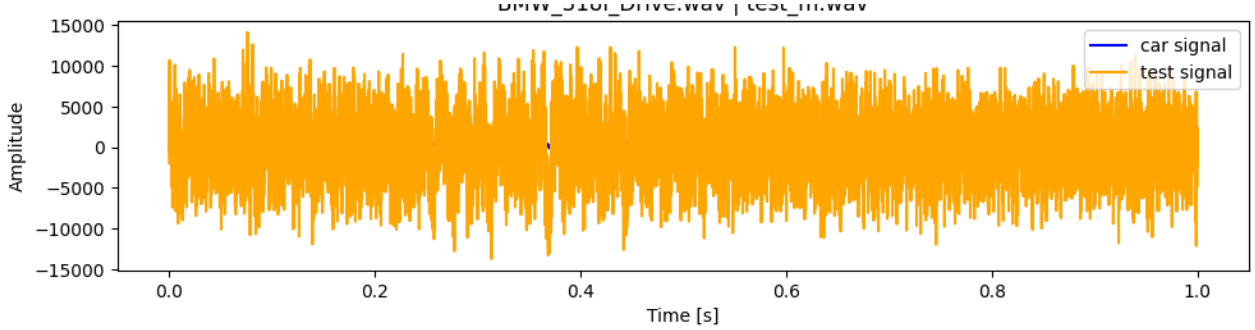


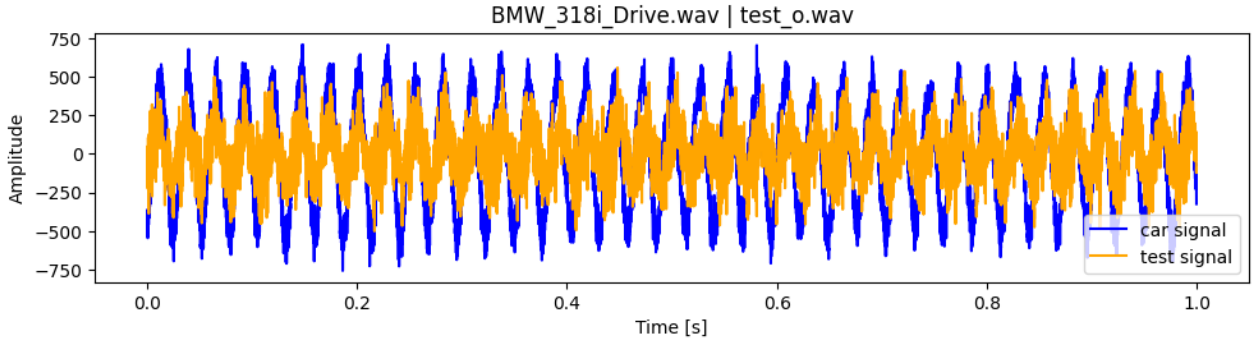


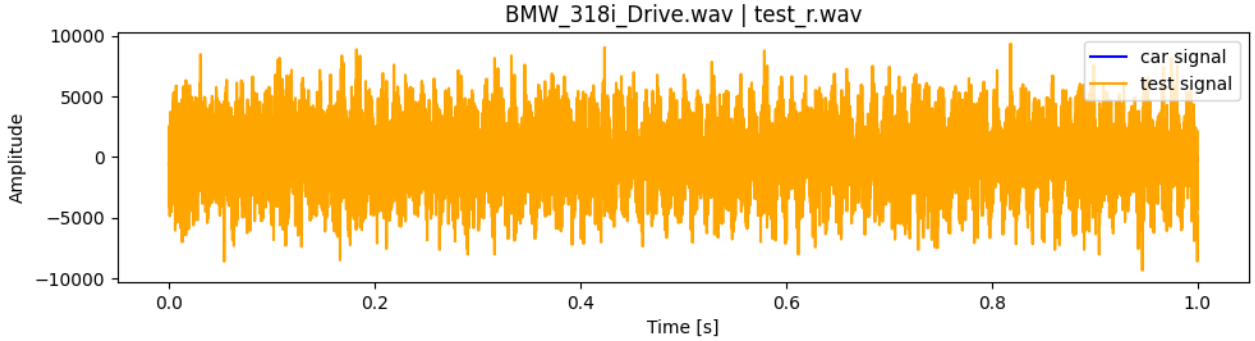


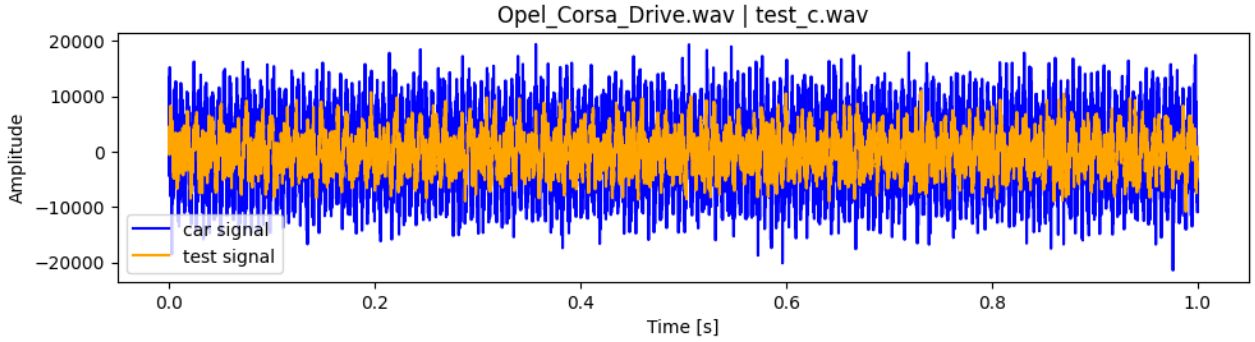


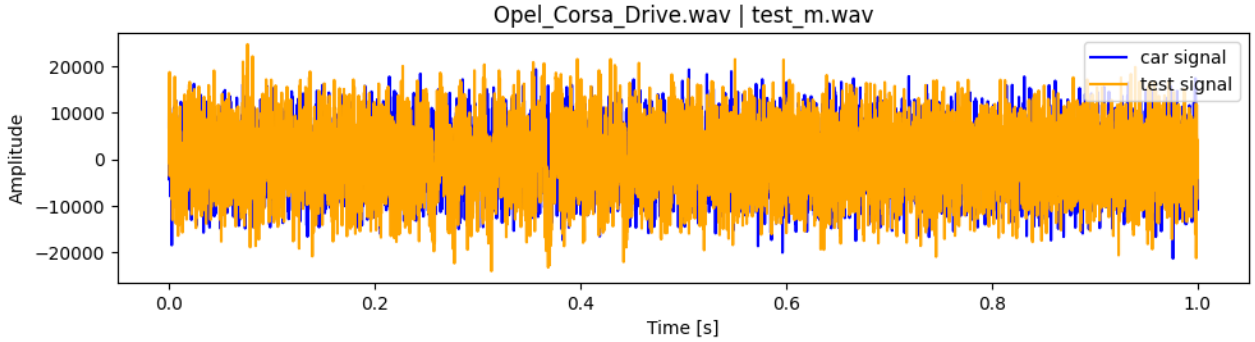
BMW_318i_Drive.wav | test_m.wav

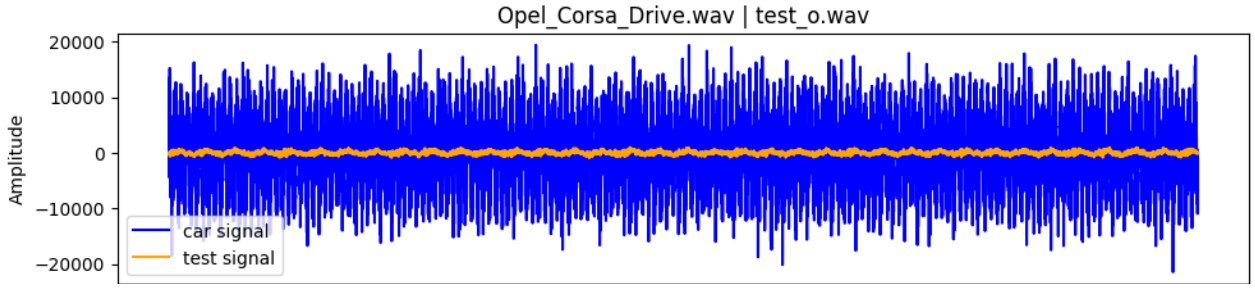

car signal
test signal
Amplitude
Time [s]

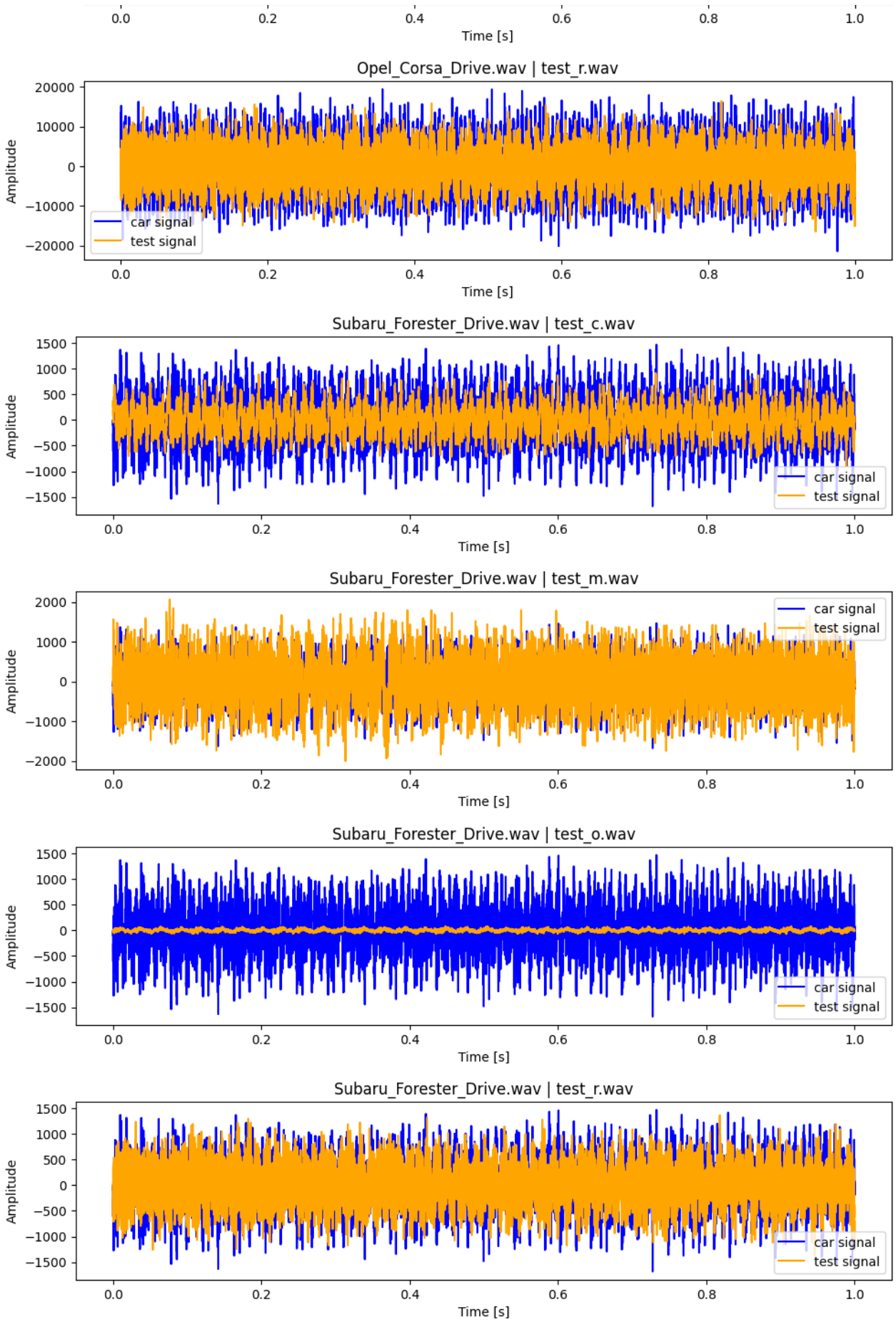

BMW_318i_Drive.wav | test_o.wav
car signal
test signal
Amplitude
Time [s]


BMW_318i_Drive.wav | test_r.wav
car signal
test signal
Amplitude
Time [s]


Opel_Corsa_Drive.wav | test_c.wav
car signal
test signal
Amplitude
Time [s]


Opel_Corsa_Drive.wav | test_m.wav
car signal
test signal
Amplitude
Time [s]


Opel_Corsa_Drive.wav | test_o.wav
car signal
test signal
Amplitude


Time [s]
Opel_Corsa_Drive.wav | test_r.wav
Amplitude
car signal
test signal
Time [s]
Subaru_Forester_Drive.wav | test_c.wav
Amplitude
car signal
test signal
Time [s]
Subaru_Forester_Drive.wav | test_m.wav
Amplitude
car signal
test signal
Time [s]
Subaru_Forester_Drive.wav | test_o.wav
Amplitude
car signal
test signal
Time [s]
Subaru_Forester_Drive.wav | test_r.wav
Amplitude
car signal
test signal
Time [s]

Možné dvojice zvukov podľa môjho názoru: (nie vysledok!)

- BMW_318i_Drive - test_o.wav
- Subaru_Forester_Drive.wav - test_c.wav
- Opel_Corsa_Drive.wav - test_m.wav
- BMW_1_Drive.wav - test_r.wav

Podľa zdanlivej podobnosti na grafe ale nemôžme určiť, či sa signály skutočne podobajú. Budem preto pokračovať v meraní pre získanie presnejších výsledkov.

2 - Fourierov rad

O fourierovom rade vieme, že nahrádza periodický signál nekonečným radom sínov a cosínov. Môj nápad je, že by som mohol uplatniť Fourierov rad na jednotlivé signály a potom ich porovnať.

Na uskutočnenie DFT používam funkciu 'fft()' a uplatním ju na každý signál. Získané dvojice transformovaných signálov a ich korelačné koeficienty načítam do pomocných zoznamov 'all_corrcoef[]' a 'all_pairs[]'. Na výstupe sú získané hodnoty.

```
car_files = ['/content/xmesikj00/BMW_1_Drive.wav','/content/xmesikj00/BMW_318i_D
test_files = ['/content/xmesikj00/test_c.wav', '/content/xmesikj00/test_m.wav',

# pomocny list pre vsetky hodnoty korelacnych koeficientov
all_corrcoef = []


# pomocny list pre vsetky mozne dvojice zvuk auta - neznamy zvuk
all_pairs = []

# tento cyklus vytvori dvojice a ich korelacny koef. a prida ich do prislusneho
for car in car_files:

  samplerate, car_sound = wavfile.read(car)
  time = np.linspace(0, len(car_sound) / samplerate, len(car_sound))

  # funkcia fft - fast fourier transform - prevedie signal na nek. rad sinov a c
  fft_car = fft(car_sound)
  for test in test_files:

    samplerate, test_sound = wavfile.read(test)
    fft_test = fft(test_sound)

    all_pairs.append(car[19:]+"|"+test[19:])
    # ziskam hodnotu korelacneho koeficientu
    x = np.correlate(fft_car, fft_test)
    all_corrcoef.append(x)


for i in range(len(all_corrcoef)):
  print("{} : {}".format(all_pairs[i], all_corrcoef[i].real))
```

```
BMW_1_Drive.wav|test_c.wav : [1.34749406e+12]
BMW_1_Drive.wav|test_m.wav : [-1.67784633e+13]
BMW_1_Drive.wav|test_o.wav : [4.9389328e+10]
BMW_1_Drive.wav|test_r.wav : [4.50374693e+12]
BMW_318i_Drive.wav|test_c.wav : [2.09547808e+11]
BMW_318i_Drive.wav|test_m.wav : [-1.14762937e+13]
BMW_318i_Drive.wav|test_o.wav : [4.29722922e+12]
BMW_318i_Drive.wav|test_r.wav : [2.94283456e+11]
Opel_Corsa_Drive.wav|test_c.wav : [4.53418389e+13]
Opel_Corsa_Drive.wav|test_m.wav : [3.18213163e+13]
Opel_Corsa_Drive.wav|test_o.wav : [8.10637424e+11]
Opel_Corsa_Drive.wav|test_r.wav : [1.09530157e+13]
Subaru_Forester_Drive.wav|test_c.wav : [1.55508656e+11]
Subaru_Forester_Drive.wav|test_m.wav : [1.13670926e+13]
Subaru_Forester_Drive.wav|test_o.wav : [-1.1089312e+10]
Subaru_Forester_Drive.wav|test_r.wav : [8.992368e+10]
```

4 - Korelacia FFT

Zo získaných dvojíc a ich hodnôt korelačných koeficientov potrebujem získať 4 najväčšie, lebo väčší korelačný koeficient znamená väčšia podobnosť signálov.

Pri porovnávaní mi ale program vrátil dvojice, v ktorých sa niektoré nahrávky opakovali viac ako dvakrát. Tento problém som vyriešil tak, že vždy keď program našiel dvojicu signálov s momentálne najvyšším korelačným koeficientom, názov referenčnej nahrávky uloží do zoznamu 'off_limits', a pri nachádzaní ďalších dvojíc bude kontrolovať, či referenčná nahrávka novej dvojice už nie je v tomto zozname. Týmto docielim to, že každá referenčná nahrávka by mala mať priradenú práve jednu unikátnu neznámu. Ak by sa stalo, že tá istá neznáma nahrávka je priradená k dvom rôznym referenčným, dvojicu s menším korelačným koeficientom by som považoval za nesprávnu a danú referenčnú nahrávku by som označil ako bez páru.

Moje predošlé riešenie zahŕňalo aj to, že som do 'off_limits' pridával aj mená neznámych súborov. Dvojice by síce boli vždy unikátne, ale v prípade jednej chybnej nahrávky by som si nič nezvyčajné nemusel všimnúť. Možno by som si akurát všimol, že jedna dvojica má výrazne nižší korelačný koeficient od ostatných a chybnú nahrávku by som odlíšil podľa tejto vlastnosti, ale nový spôsob mi príde o niečo spoľahlivejší a očividnejší aj keď v mojom prípade, výsledok z oboch riešení bol rovnaký.

```
# pomocny list pre mena suborov referencnych nahravok
off_limits = []

#pocet najvacsich hodnot
num = 0

while(True):
  if num == 4:
    break

  # index najvacsieho prvku (a prislusnej dvojice signalov)
  max_index = all_corrcoef.index(max(all_corrcoef))

  curr_car = all_pairs[max_index][:all_pairs[max_index].find("|")]

  # podmienka, ktora kontroluje, aby sa ref. nahravky neopakovali
  if (curr_car not in off_limits) :

    print(all_pairs[max_index], end=" : ")
    print(all_corrcoef[max_index])

    # pridam nazov ref. nahravky do pomocneho zoznamu
    off_limits.append(curr_car)

    # odstranim najvacsie hodnoty
    all_pairs.pop(max_index)
```

```
    all_pairs.pop(max_index)
    all_corrcoef.pop(max_index)
    num += 1
  else:
    # ak by sa mal pouzit signal, ktory sme si uz zobrazili, odstranime ho zo z
    all_pairs.pop(max_index)
    all_corrcoef.pop(max_index)
```

```
Opel_Corsa_Drive.wav|test_c.wav : [4.53418389e+13-0.02539062j]
Subaru_Forester_Drive.wav|test_m.wav : [1.13670926e+13-0.00878906j]
BMW_1_Drive.wav|test_r.wav : [4.50374693e+12+0.00488281j]
BMW_318i_Drive.wav|test_o.wav : [4.29722922e+12+0.00244141j]
```

Z výsledkov si môžme všimnúť, že žiadna neznáma nahrávka nie je priradená 2krát, a teda každá ref. nahrávka má priradenú unikátnu neznámu nahravku. Okrem toho si môžme všimnúť, že reálna zložka korelačných koeficientov prvých dvoch dvojíc je od zvyšných o niečo vyššia, ale žiadna dvojica sa výraznejšie nelíši od ostatných. Preto budem uvažovať získané hodnoty za správne, a teda, že každá referenčná nahrávka mojej sady má pár.

5 - Výsledok

- BMW_318i_Drive - test_o.wav
- Subaru_Forester_Drive.wav - test_m.wav
- Opel_Corsa_Drive.wav - test_c.wav
- BMW_1_Drive.wav - test_r.wav

zdroje:

https://matplotlib.org/stable

https://docs.scipy.org/doc/scipy/tutorial/fft.html

https://www.fit.vut.cz/study/course/ISS/public/opora/iss.pdf