ISS projekt 2024/25 "Podle hlasu poznáte je ..."

Autor: Radim Dvořák xdvorar00

Datum: 16.12.2024

```
# 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

# read the file - change login to YOUR login
login = "xdvorar00"
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
```

```
--2024-12-16 22:12:24--
https://www.fit.vut.cz/study/course/ISS/public/proj2024-25/personal/
xdvorar00.zip
Resolving www.fit.vut.cz (www.fit.vut.cz)... 147.229.9.65,
2001:67c:1220:809::93e5:941
Connecting to www.fit.vut.cz (www.fit.vut.cz)|147.229.9.65|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 217334 (212K) [application/zip]
Saving to: ‘xdvorar00.zip.90’

xdvorar00.zip.90    100%[===================>] 212.24K   232KB/s    in
0.9s

2024-12-16 22:12:25 (232 KB/s) - ‘xdvorar00.zip.90’ saved
[217334/217334]

Archive:  xdvorar00.zip
```

```
  inflating: xdvorar00/BMW_318i_Drive.wav
  inflating: xdvorar00/test_i.wav
  inflating: xdvorar00/Audi_A3_Drive.wav
  inflating: xdvorar00/test_n.wav
  inflating: xdvorar00/BMW_1_Drive.wav
  inflating: xdvorar00/test_r.wav
  inflating: xdvorar00/Mercedes_300SE_Drive.wav
  inflating: xdvorar00/test_d.wav
```

```
# 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)
```

```
----- test signals ---------
['test_d', 'test_r', 'test_i', 'test_n']
```

```
<IPython.lib.display.Audio object>
```

```
<IPython.lib.display.Audio object>
```

```
<IPython.lib.display.Audio object>
```

```
<IPython.lib.display.Audio object>
```

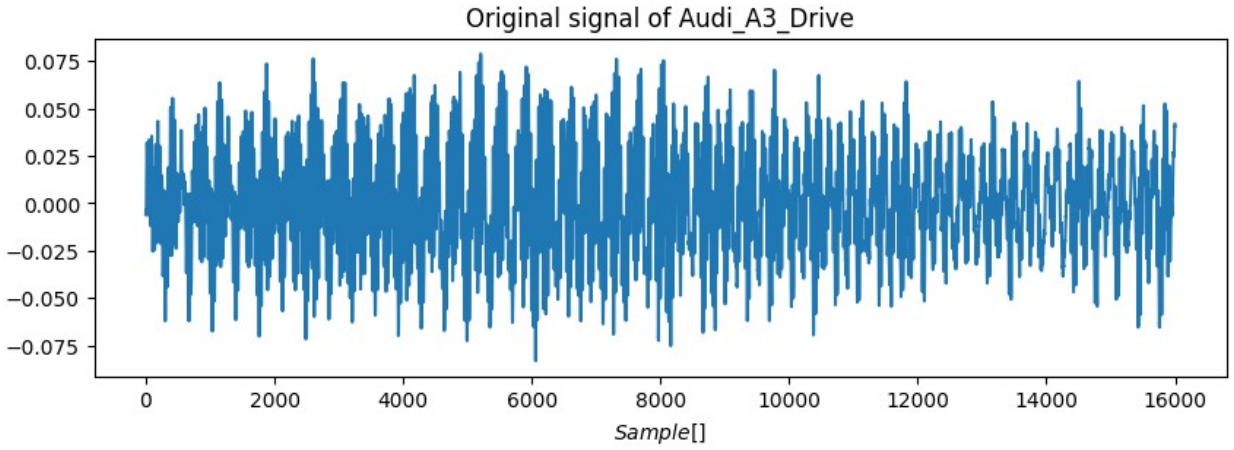
```
----- reference signals ---------
['Audi_A3_Drive', 'BMW_318i_Drive', 'BMW_1_Drive',
'Mercedes_300SE_Drive']

<IPython.lib.display.Audio object>

<IPython.lib.display.Audio object>

<IPython.lib.display.Audio object>

<IPython.lib.display.Audio object>
```

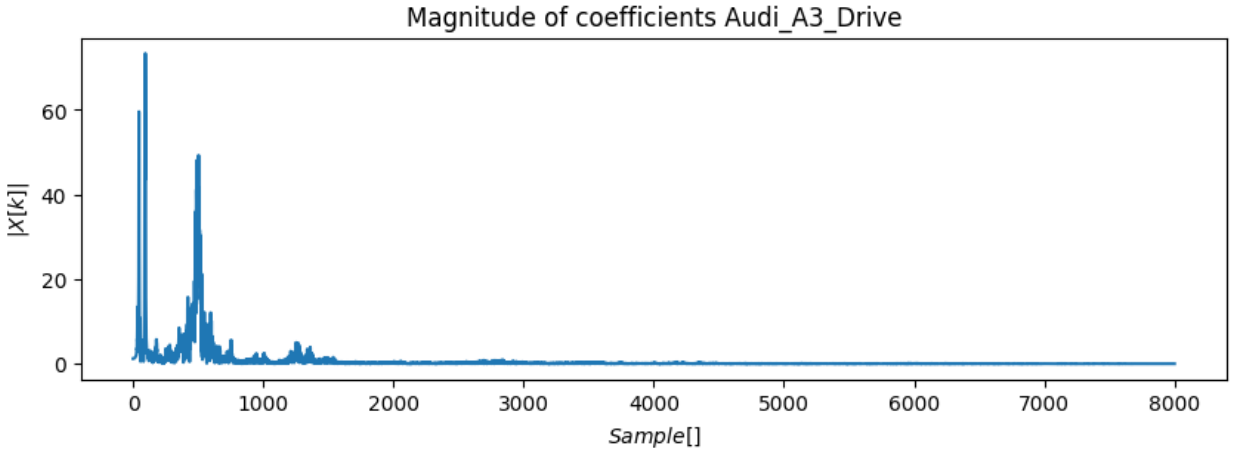
Diskrétní Fourierova transformace

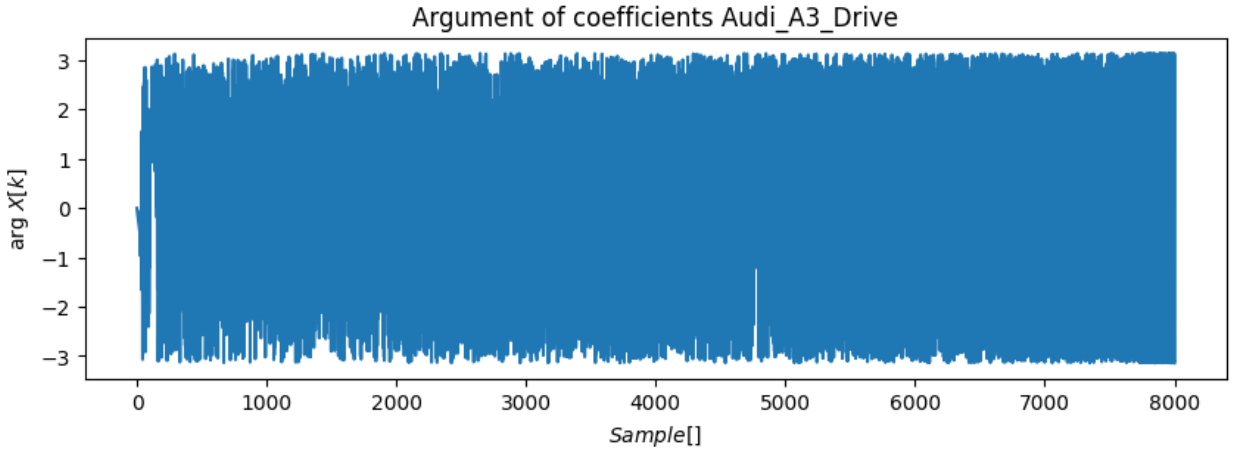
Nejprve si zobrazím refereční signály na časové doméně a zjistím si koeficienty DFT. Koeficienty budou uloženy v polích a to magnituda a argument ve samostatných polích. Tyto koeficienty nám stačí zjistit do N/2 vzorků, další už budou komplexně združené už dříve zjištěným. A pro každý signál si vytvořím spektrální diagram.

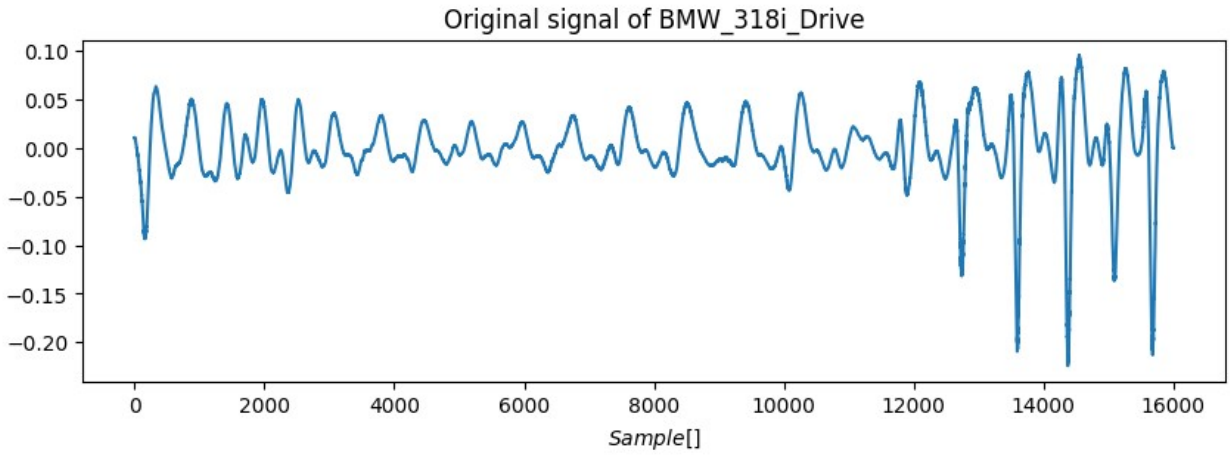
```
N = Fs
n = np.arange(0,N)

ref_dft = np.empty(N_ref,dtype=object)
ref_dft_mag = np.empty(N_ref,dtype=object)
ref_dft_arg = np.empty(N_ref,dtype=object)

# Pro každý referenční signál udělej DFT
for ref_index in range(N_ref):
  plt.figure(figsize=(10,3))
  plt.plot(n, ref_signals[ref_index])

  ref_dft[ref_index] = np.fft.fft(ref_signals[ref_index])

  # Stačí jenom N/2 vzorků
  kall = np.arange(0,int(N/2) +1)

  # Získání magnitudy a argumentu koeficintů
  ref_dft_mag[ref_index] = np.abs(ref_dft[ref_index][kall])
  ref_dft_arg[ref_index] = np.angle(ref_dft[ref_index][kall])

  plt.title("Original signal of " + ref_labels[ref_index])
  plt.xlabel('$Sample []$')

  plt.figure(figsize=(10,3))
  plt.title("Magnitude of DFT coefficients " + ref_labels[ref_index])
  plt.ylabel('$|X[k]|$')
  plt.xlabel('$Sample []$')
  plt.plot(kall,ref_dft_mag[ref_index])
```

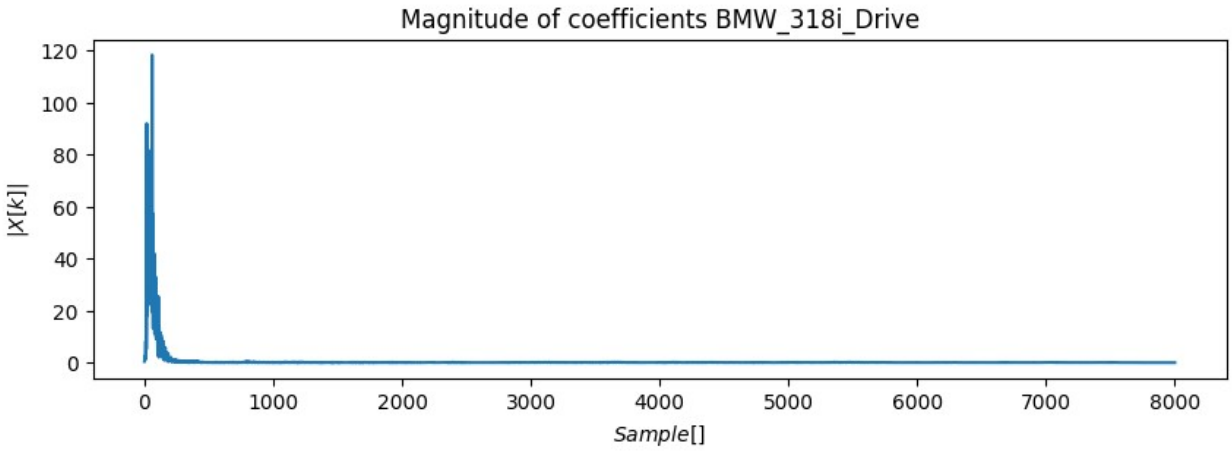
```
plt.figure(figsize=(10,3))
plt.title("Argument of DFT coefficients " + ref_labels[ref_index])
plt.ylabel('arg $X[k]$')
plt.xlabel('$Sample []$')
plt.plot(kall,ref_dft_arg[ref_index])
```

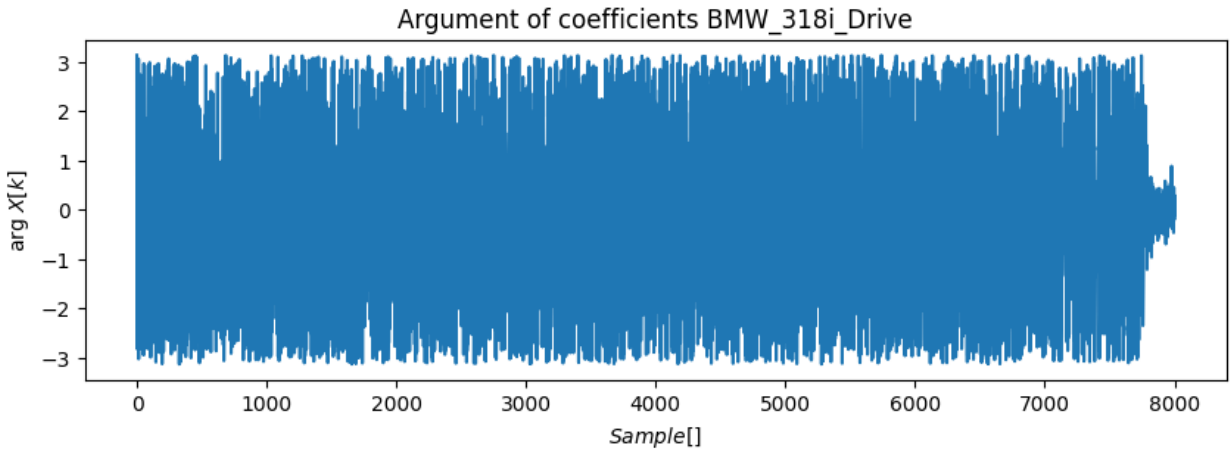


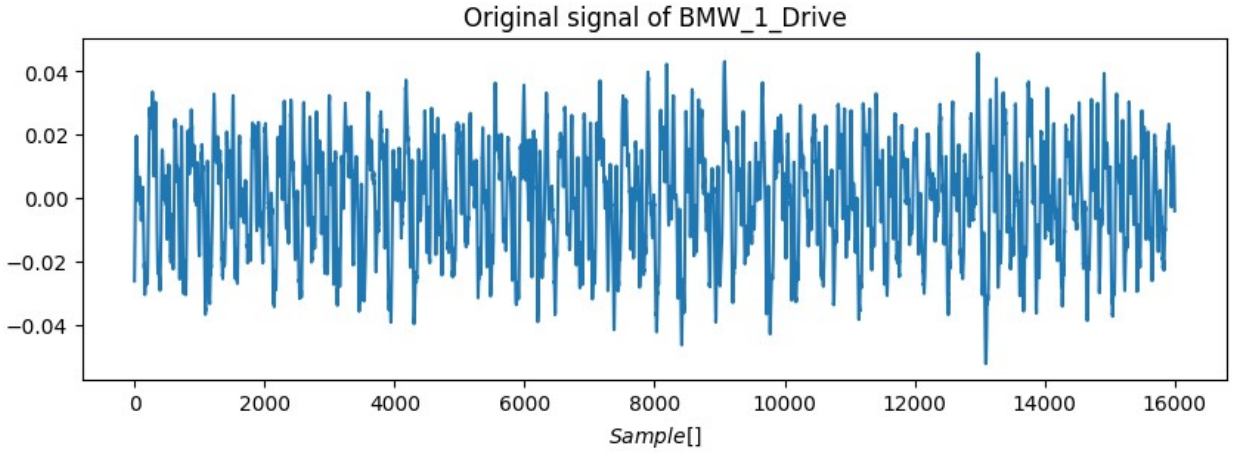


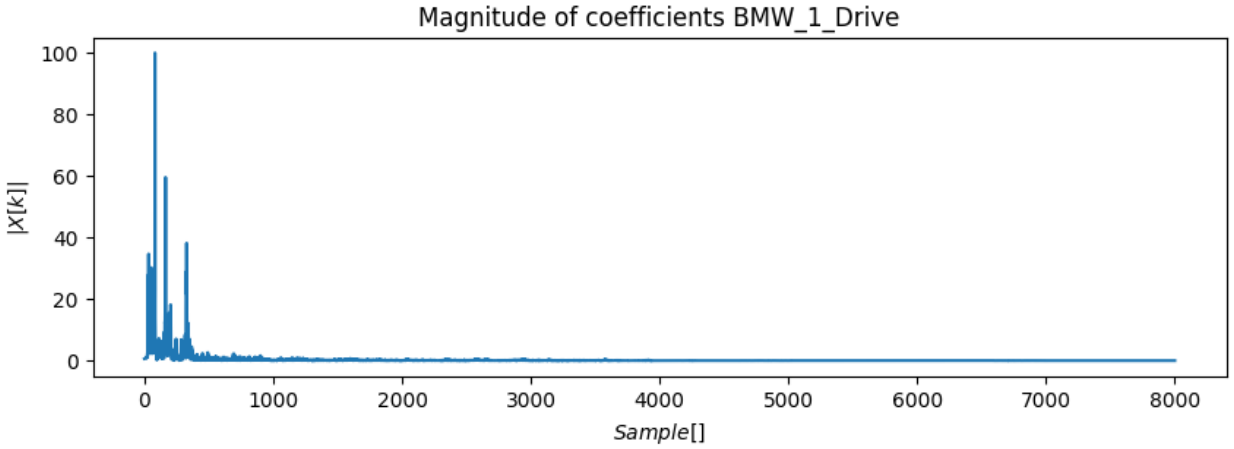


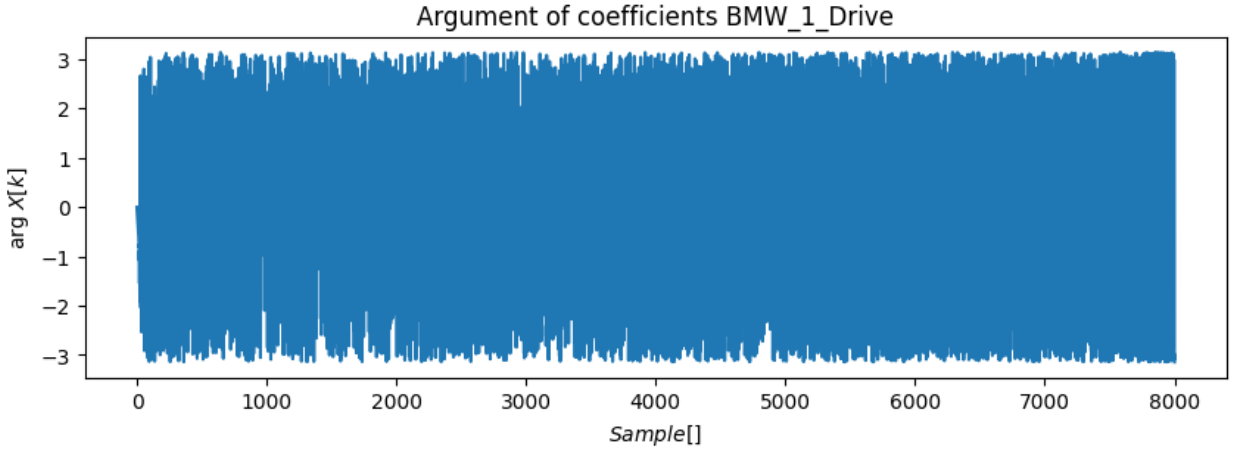

Original signal of BMW_318i_Drive
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
0
2000
4000
6000
8000
10000
12000
14000
16000
Sample[]

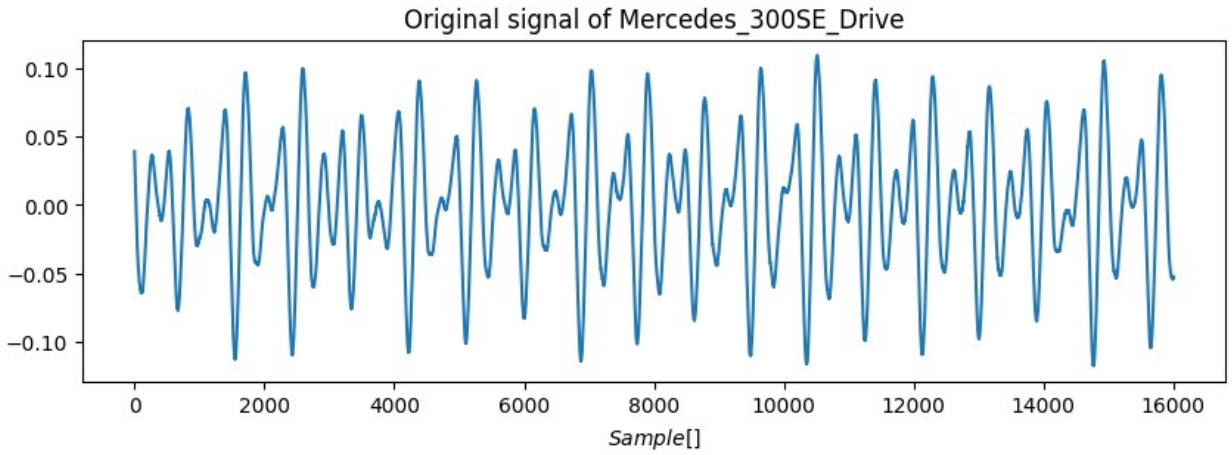

Magnitude of coefficients BMW_318i_Drive
|X[k]|
120
100
80
60
40
20
0
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]

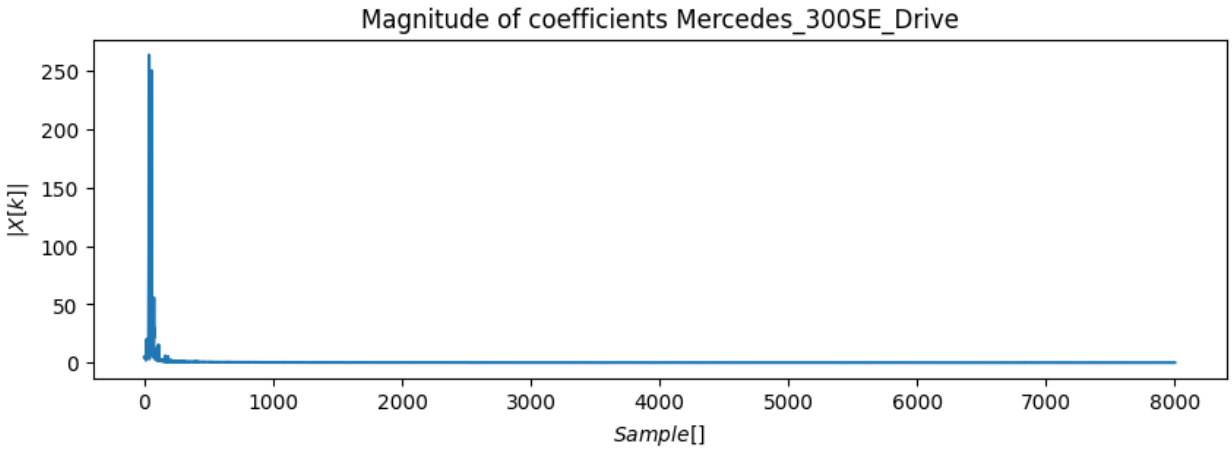

Argument of coefficients BMW_318i_Drive
arg X[k]
3
2
1
0
−1
−2
−3
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]

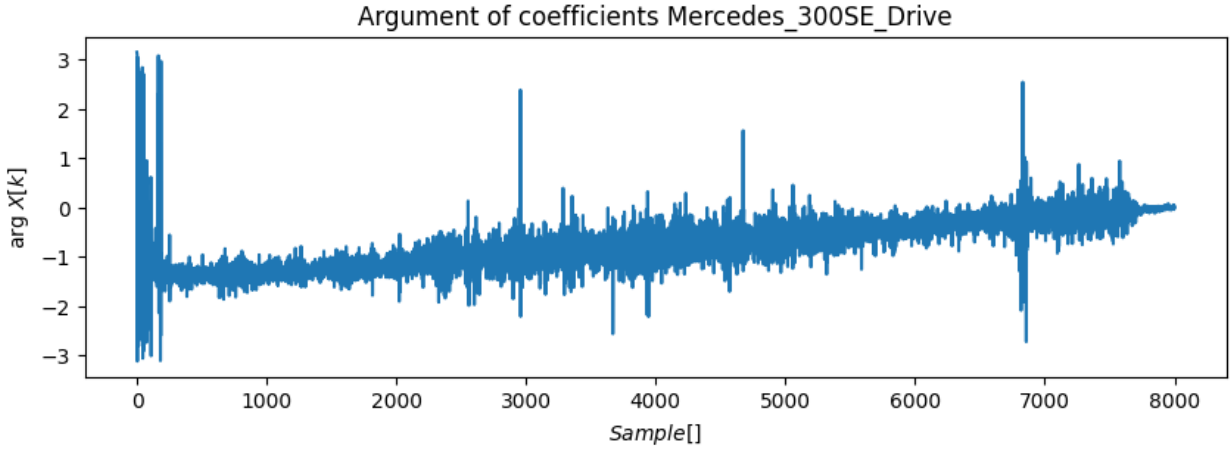

Original signal of BMW_1_Drive
0.04
0.02
0.00
−0.02
−0.04
0
2000
4000
6000
8000
10000
12000
14000
16000
Sample[]


Magnitude of coefficients BMW_1_Drive
|X[k]|
100
80
60
40
20
0
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]


Argument of coefficients BMW_1_Drive
arg X[k]
3
2
1
0
−1
−2
−3
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]



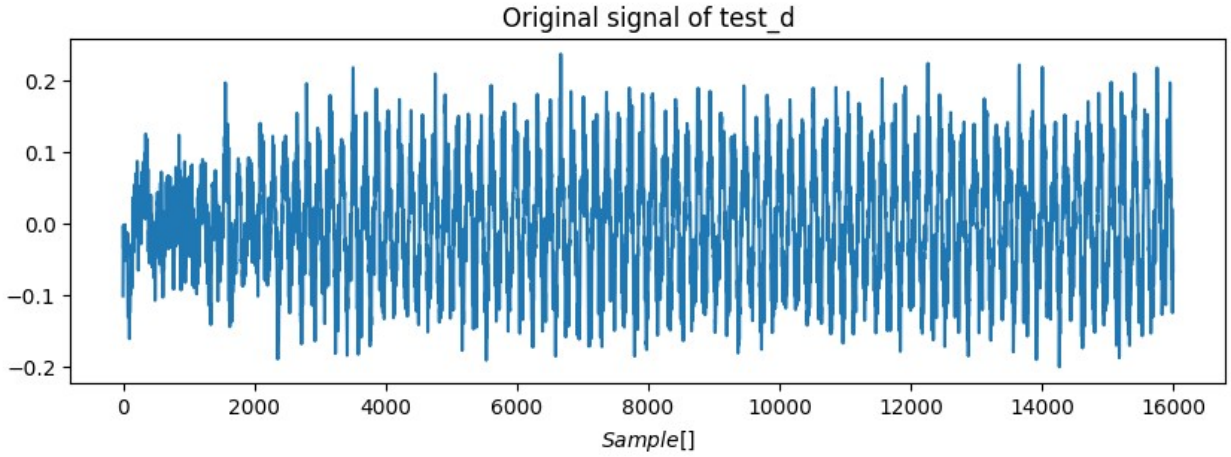


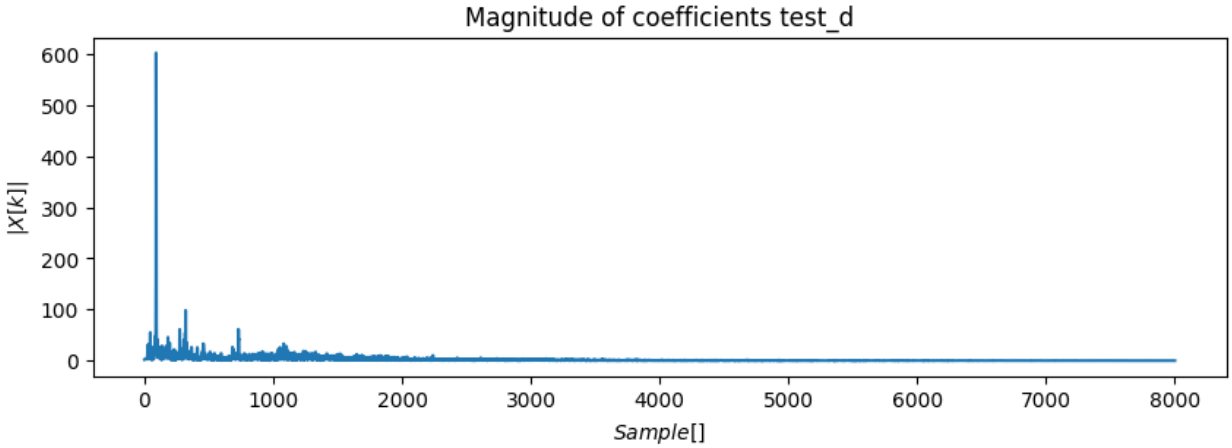


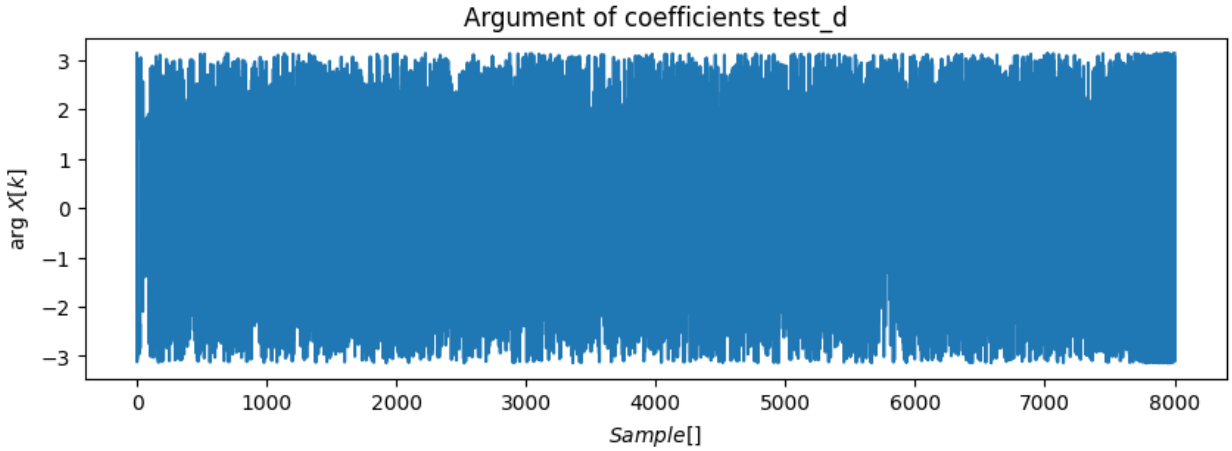
Úplně stejný postup použiji na testované signály.

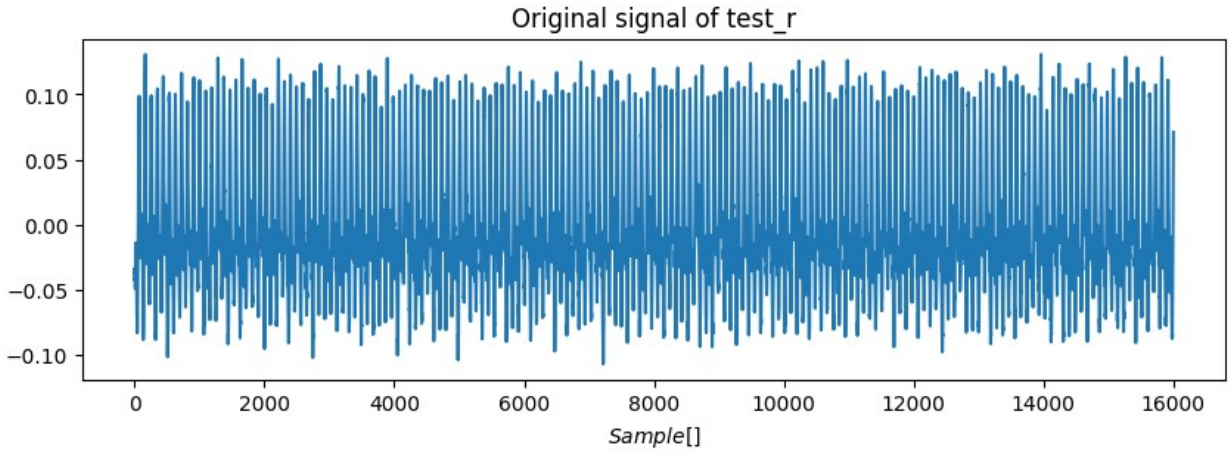
```
test_dft = np.empty(N_ref,dtype=object)
test_dft_mag = np.empty(N_test,dtype=object)
test_dft_arg = np.empty(N_test,dtype=object)
```

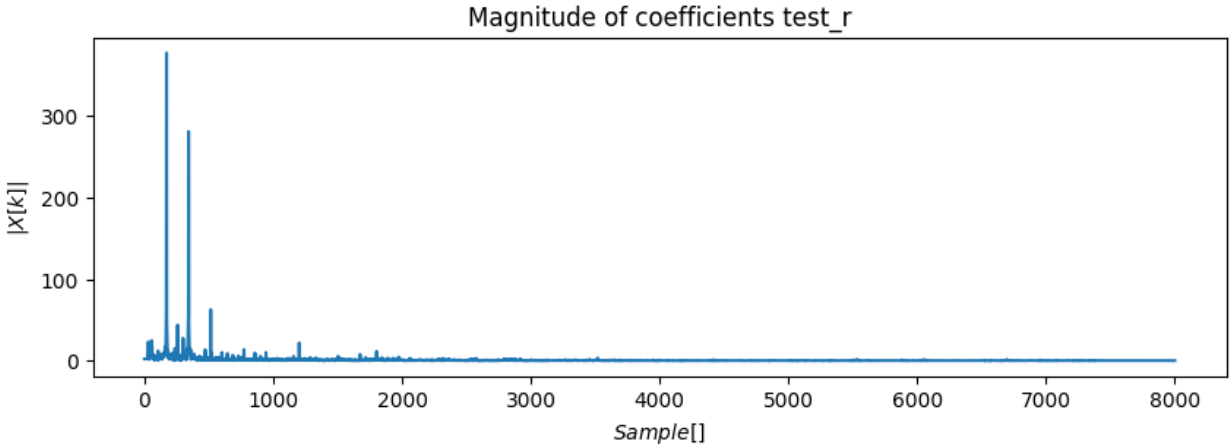
```
# Pro každý testovací signál udělej DFT
for test_index in range(N_test):
  plt.figure(figsize=(10,3))
  plt.plot(n, test_signals[test_index])

  test_dft[test_index] = np.fft.fft(test_signals[test_index])

  # Stačí jenom N/2 vzorků
  kall = np.arange(0,int(N/2) +1)

  # Získání magnitudy a argumentu koeficintů
  test_dft_mag[ref_index] = np.abs(test_dft[test_index][kall])
  test_dft_arg[ref_index] = np.angle(test_dft[test_index][kall])

  plt.title("Original signal of " + test_labels[test_index])
  plt.xlabel('$Sample []$')

  plt.figure(figsize=(10,3))
  plt.title("Magnitude of DFT coefficients " +
test_labels[test_index])
  plt.ylabel('$|X[k]|$')
  plt.xlabel('$Sample []$')
  plt.plot(kall,test_dft_mag[ref_index])

  plt.figure(figsize=(10,3))
  plt.title("Argument of DFT coefficients " + test_labels[test_index])
  plt.ylabel('arg $X[k]$')
  plt.xlabel('$Sample []$')
  plt.plot(kall,test_dft_arg[ref_index])
```

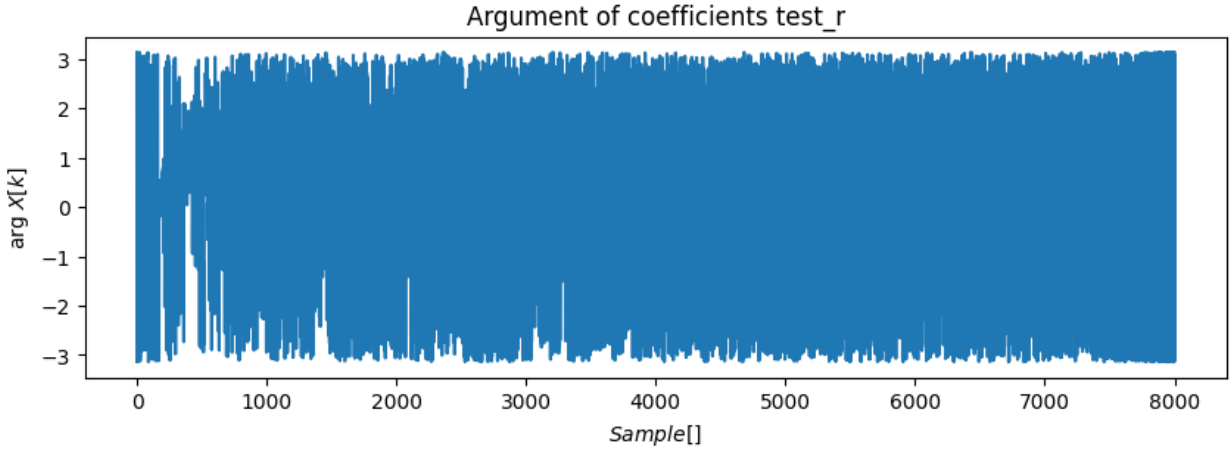


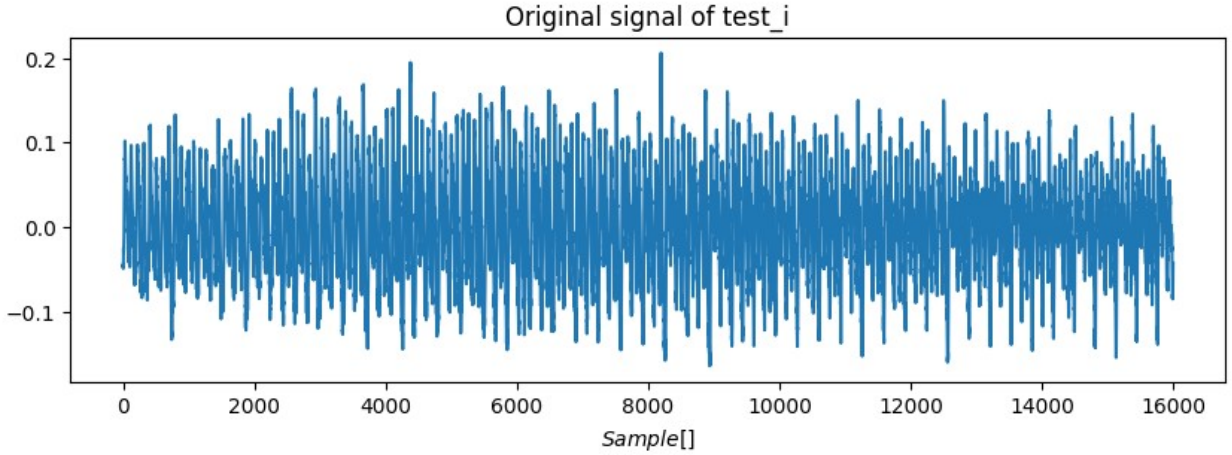

Magnitude of coefficients test_d
|X[k]|
600
500
400
300
200
100
0
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]

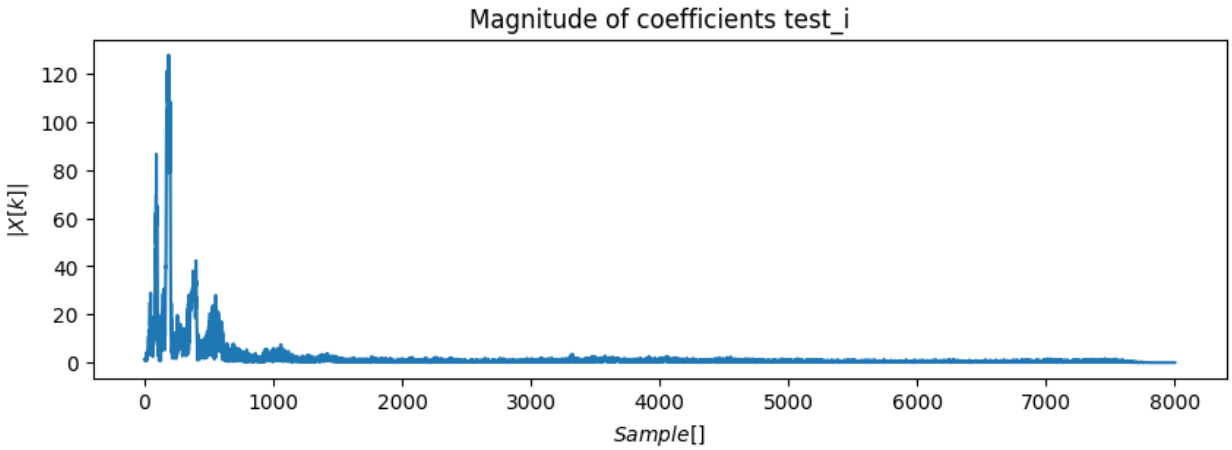

Argument of coefficients test_d
arg X[k]
3
2
1
0
−1
−2
−3
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]

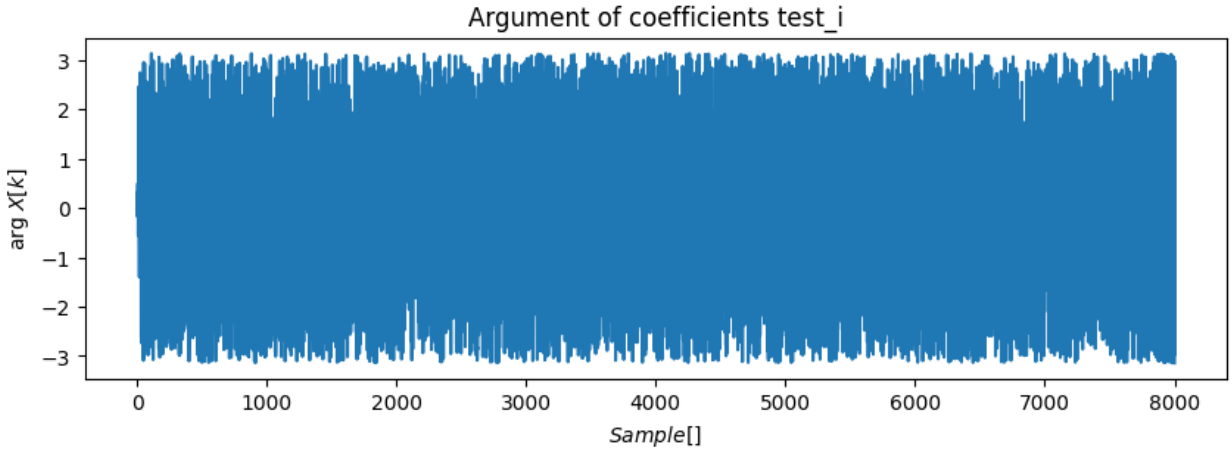

Original signal of test_r
0.10
0.05
0.00
−0.05
−0.10
0
2000
4000
6000
8000
10000
12000
14000
16000
Sample[]

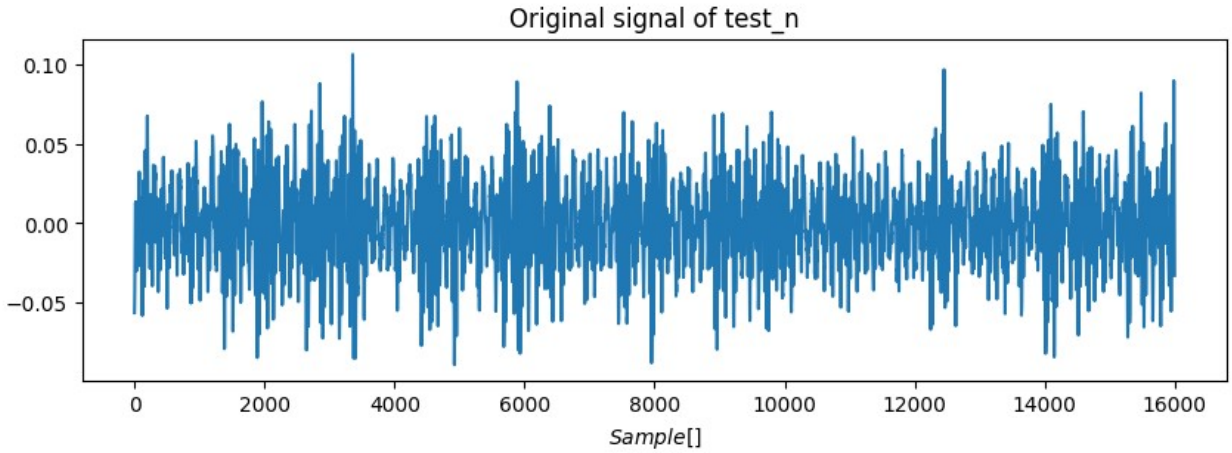

Magnitude of coefficients test_r
|X[k]|
300
200
100
0
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]

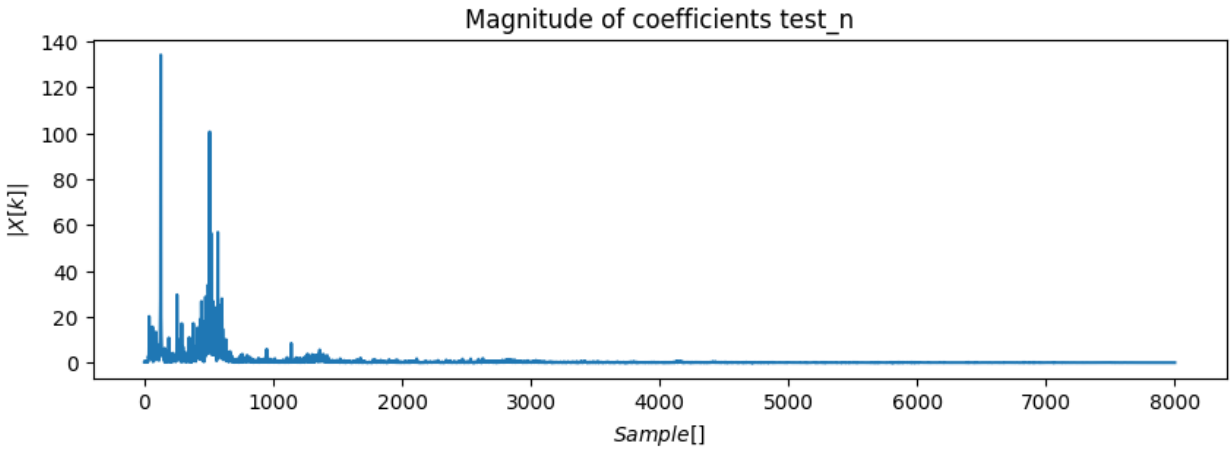

Argument of coefficients test_r
arg X[k]
3
2
1
0
−1
−2
−3
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]

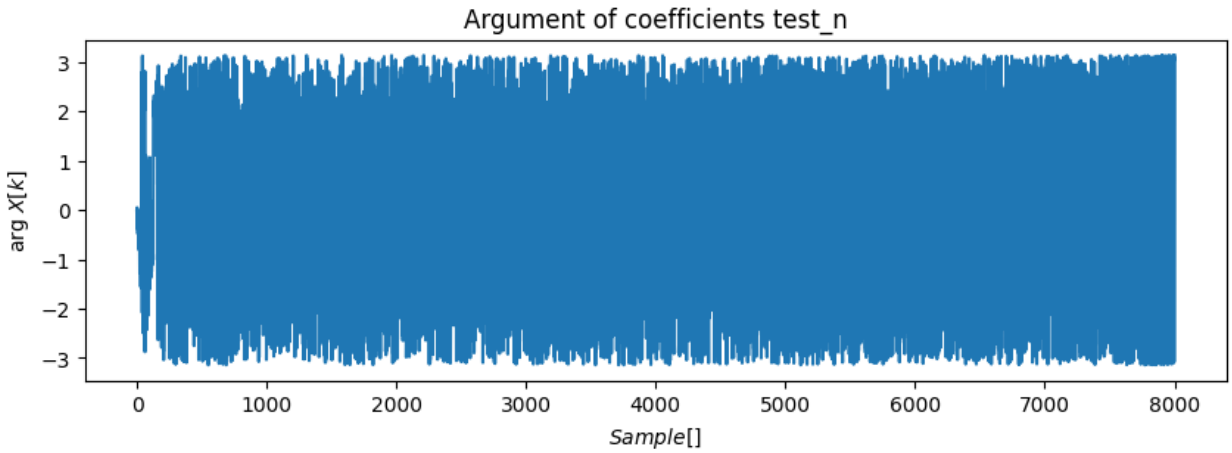

Original signal of test_i
0.2
0.1
0.0
−0.1
0
2000
4000
6000
8000
10000
12000
14000
16000
Sample[]


Magnitude of coefficients test_i
|X[k]|
120
100
80
60
40
20
0
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]


Argument of coefficients test_i
arg X[k]
3
2
1
0
−1
−2
−3
0
1000
2000
3000
4000
5000
6000
7000
8000
Sample[]


Original signal of test_n
0.10
0.05
0.00
−0.05
0
2000
4000
6000
8000
10000
12000
14000
16000
Sample[]





Frekveční doména

V této části zkombinuju násobění koeficientů DFT testovacího signálu a známého signálu a a vynásobení vzorků obou signálů a následné provedení DFT. Oba způsoby budou váhově ohodnoceny Tímto získáme podobnost ve frekveční doméně.

```
#weighs for calculating weighted coefficient
w1 = .8
w2 = .7
for test_index in range(N_test):
  corr_index = 0
  for ref_index in range(N_ref):
    prev_corr = 0
    corr = w1 * sum(test_dft[test_index] * ref_dft[ref_index]) + w2 *
sum(test_signals[test_index] * ref_signals[ref_index])
    if corr > prev_corr:
      corr_index = ref_index
      prev_corr = corr
```

```
  print("Test sample: " + test_labels[test_index] + " Reference
sample: " + ref_labels[corr_index] + " Similarity:",abs(corr))
```

```
Test sample: test_d Reference sample: BMW_318i_Drive Similarity:
13565.797480192652
Test sample: test_r Reference sample: BMW_318i_Drive Similarity:
1739.8864989906517
Test sample: test_i Reference sample: Mercedes_300SE_Drive Similarity:
2463.1724542112975
Test sample: test_n Reference sample: Mercedes_300SE_Drive Similarity:
9178.395137480304
```

Dynamic time warping

Z výsledků je patrné že toto není velmi efektivní metoda, protože přiřazuje ke dvou testovacím vzorkům stejný známý signál. Jako další algoritmus zvaný Dynamic time warping(DTW), které dokáže posoudit zda dva signály se liší v rychlosti. Pro každý testovací signál porovnám pomocí tohoto algoritmu každý známý signál.

```
!pip install fastdtw
from fastdtw import fastdtw

for test_index in range(N_test):
  corr_index = 0
  for ref_index in range(N_ref):
    prev_corr = 0
    distance, _ =
fastdtw(ref_signals[ref_index],test_signals[test_index])
    if distance < prev_corr:
      corr_index = ref_index
      prev_corr = distance
  print("Test sample: " + test_labels[test_index] + " Reference
sample: " + ref_labels[corr_index] + " Distance:",distance)
```

```
Requirement already satisfied: fastdtw in
/usr/local/lib/python3.10/dist-packages (0.3.4)
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (from fastdtw) (1.26.4)
Test sample: test_d Reference sample: Audi_A3_Drive Distance:
860.1330871582031
Test sample: test_r Reference sample: Audi_A3_Drive Distance:
575.8416748046875
Test sample: test_i Reference sample: Audi_A3_Drive Distance:
744.1375122070312
Test sample: test_n Reference sample: Audi_A3_Drive Distance:
403.6739501953125
```

Závěr

S použitými dvěma metodami jsem nedokázal účinně dokázat, jaké nahrávky patří k sobě. A nedokázal jsem ani odhalit jestli je nějaká nenáleží k známé nahrávce.

Seznam použité literatury

[1] https://stackoverflow.com/questions/20644599/similarity-between-two-signals-looking-for-simple-measure

[2] https://en.wikipedia.org/wiki/Dynamic_time_warping

[3] https://dsp.stackexchange.com/questions/76673/what-algorithm-can-i-use-to-compare-two-signals-similarity

[4] Hlasy v mojí hlavě