KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

TAIKOMOSIOS INFORMATIKOS KATEDRA

Individuali egzamino užduotis
Nr. 3

Atliko:
IFK-8 grupės stud.
Tomas Odinas

Priėmė
lekt. Andrius Kriščiūnas

KAUNAS, 2021

TURINYS

1. UŽDUOTIS

Aproksimavimas

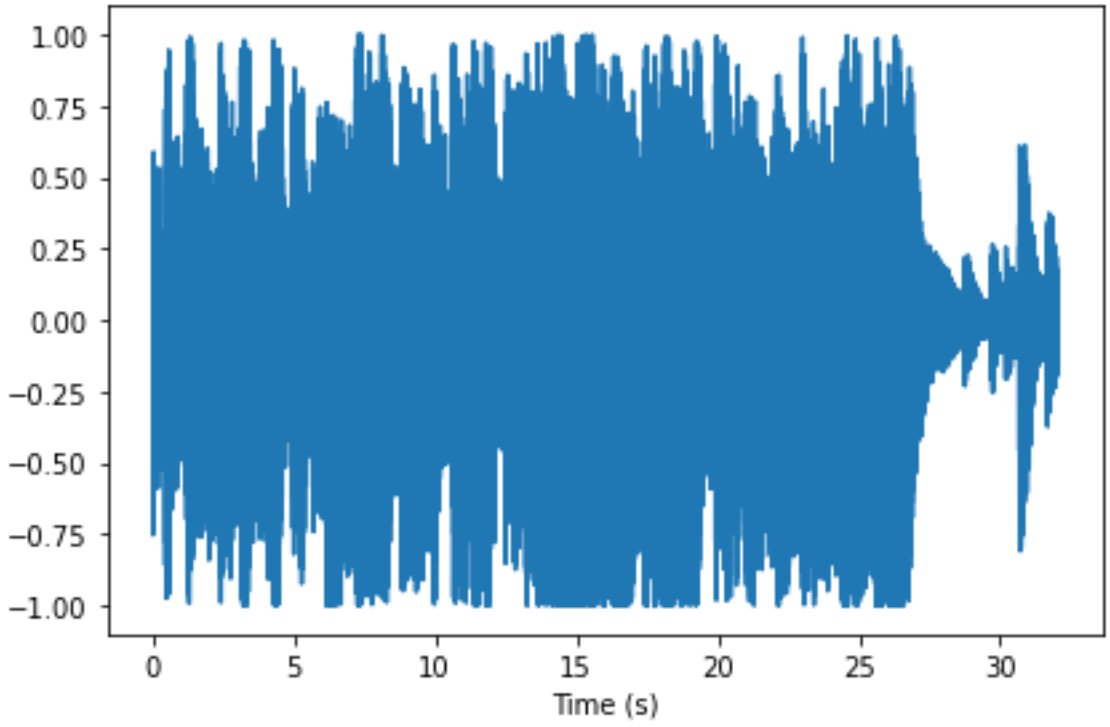
3 **Aproksimavimas**. Pasirinkto garso (WAV formatu) analizė ir filtravimas (pvz. pasirinktai dainai ar jos fragmentui, pabandyti garsą išskirti į žodžius ir muziką).

Pav. 1 Užduoties sąlyga

2. PAGRINDINĖ DALIS

Nagrinėsime „David Guetta – Titanium“[1] dainos fragmentą. Bus naudojama „Think DSP“[2] biblioteka skirta apdoroti skaitmeniniams signalams. Naudojamas FFT metodas („Fast Fourier transform“), kuris veikia analogiškai DFT metodui („Discrete Fourier transform“). Pagrindinis skirtumas jog skaičiuojant DFT metodu algoritmo sudėtingumas yra $O(n^2)$, o skaičiuojant FFT metodu sudėtingumas pagerinamas iki $O(n \log n)$.

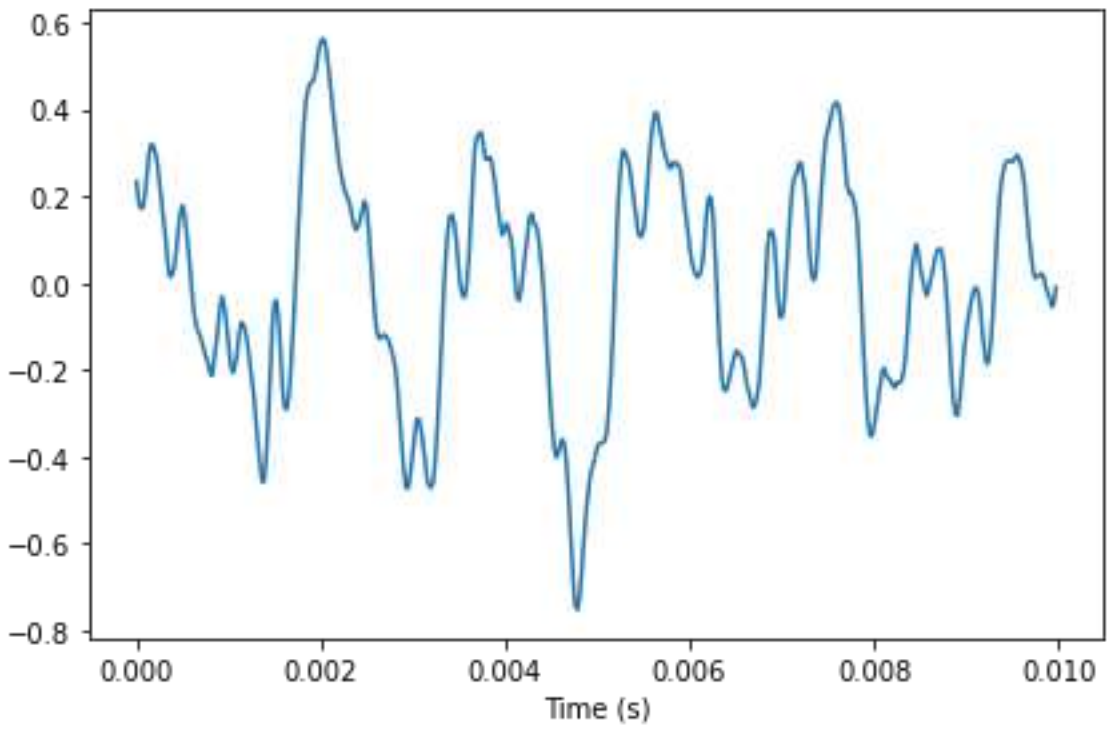
Užkrovus „wave“ failą matome kaip kinta signalas.



Pav. 2 Nagrinėjamas signalas

[1] https://www.youtube.com/watch?v=OA4fdiOqNMw
[2] https://github.com/AllenDowney/ThinkDSP

Priartinus mažą fragmentą matome, jog ši banga gali būti nagrinėjama kaip bet kokia kita matematinė funkcija.



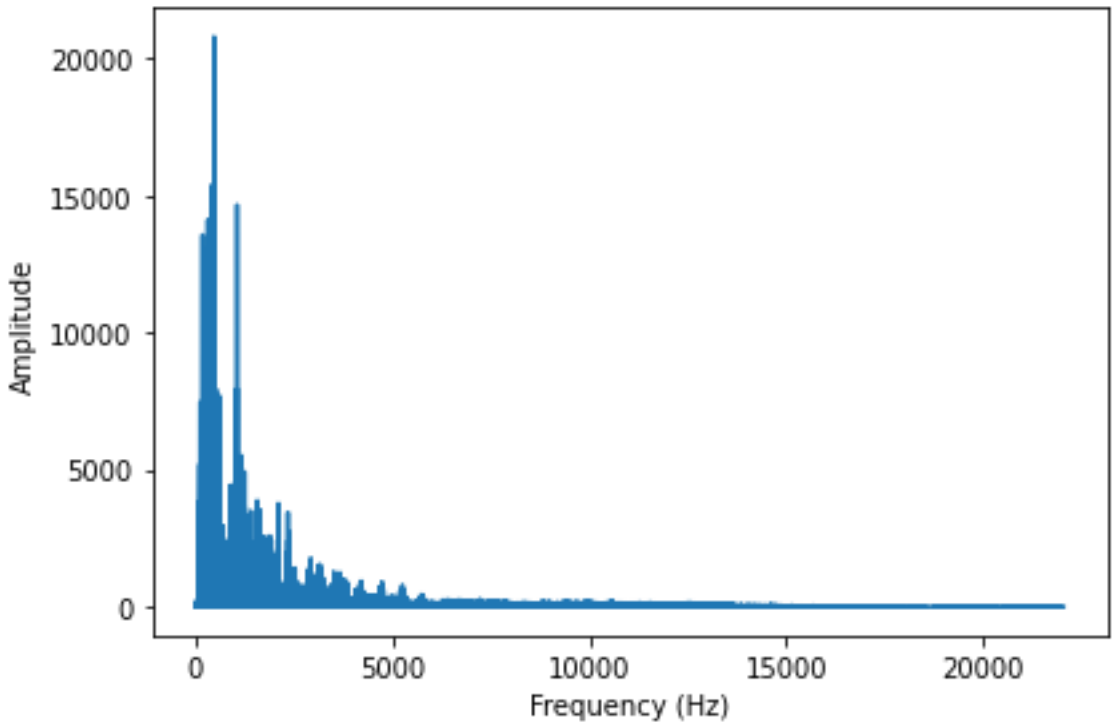
Pav. 3 Nagrinėjamo signalo mažas fragmentas

Šis signalas bus aproksimuojamas naudojant trigonometrinių bazinių funkcijų bazę (nors galima naudoti ir eksponentes su kompleksiniais skaičiais).

$$\left[1 \quad \cos\frac{2\pi t}{T} \quad \cos 2\frac{2\pi t}{T} \quad \ldots \quad \cos(m-1)\frac{2\pi t}{T} \quad \sin\frac{2\pi t}{T} \quad \sin 2\frac{2\pi t}{T} \quad \ldots \quad \sin(m-1)\frac{2\pi t}{T}\right]$$

Pav. 4 Trigonometrinių funkcijų bazė

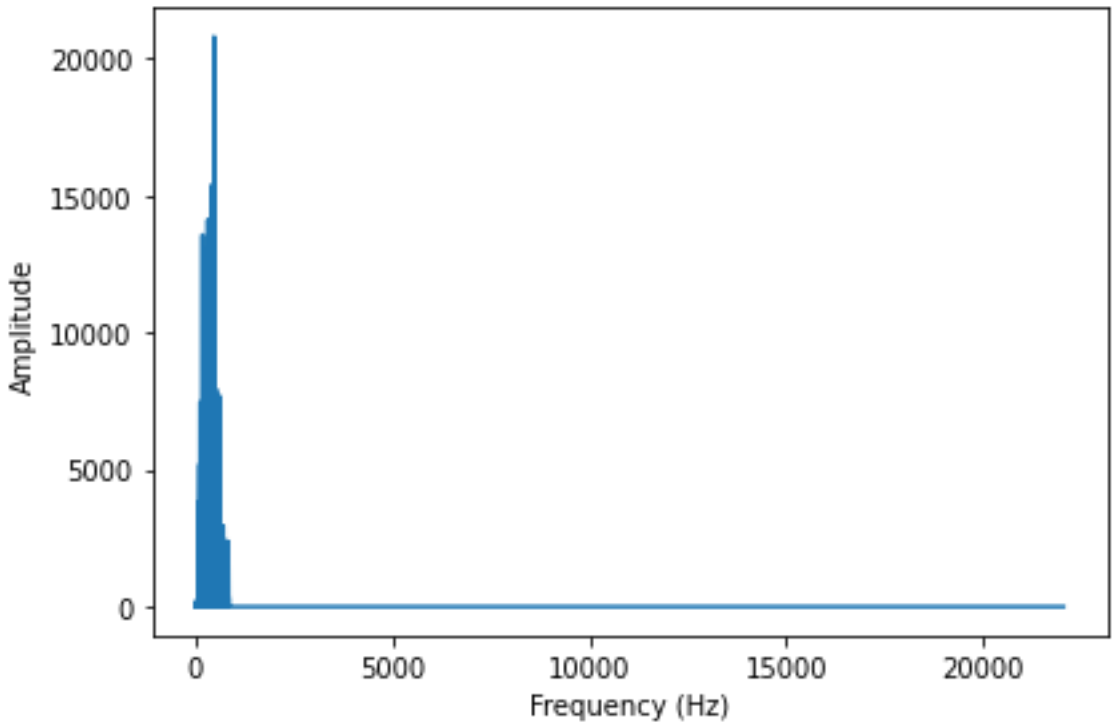
Radę koeficientus bazinėms funkcijoms gauname aproksimuojančią funkciją. Svarbu pabrėžti, jog fizikine prasme gauti koeficientai atitinka harmonikų amplitudes. Gautus dažnius ir jų amplitudes galime pavaizduoti grafiškai.



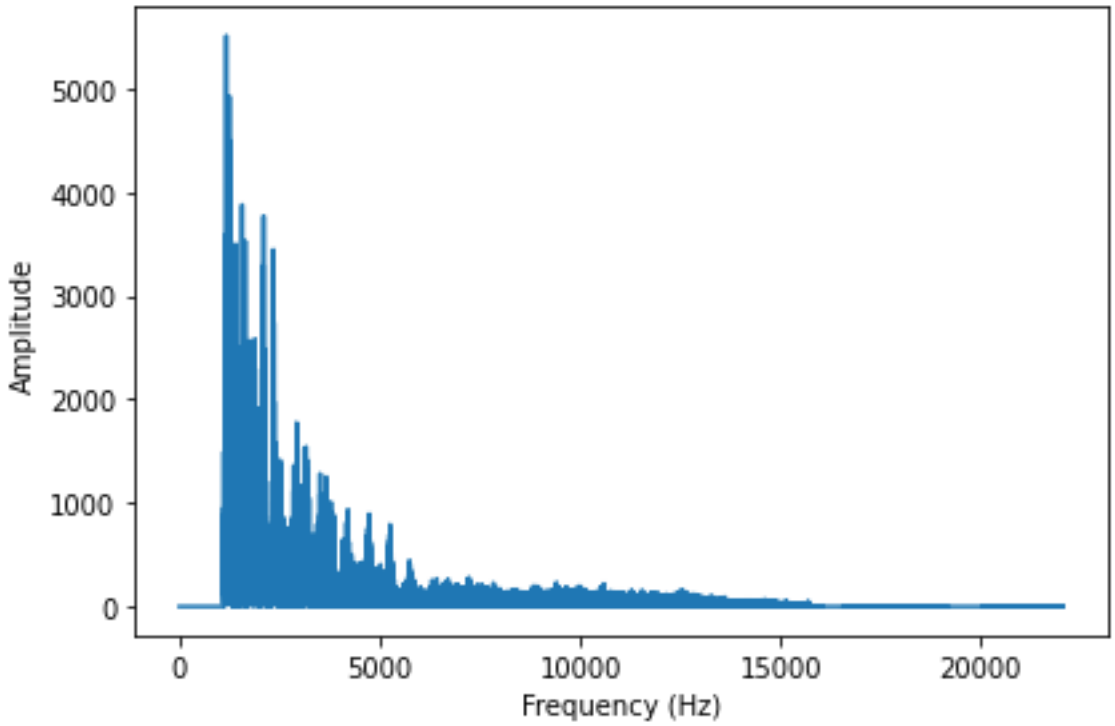
Pav. 5 Nagrinėjamo signalo dažniai ir amplitudės išskirti FFT metodu

Turėdami signalą išskirtą į dažnius ir amplitudes galime atlikti kokį tik norime jų filtravimą ar sustiprinimą ir pasidaryti muzikos ekvalaizerį arba atlikti kompresiją. Šiuo atveju atfiltruosime žemesnius dažnius, nes audio įraše merginos balsas yra aukštas lyginant su pianino muzika. Dažniai atfiltruojami jų amplitudes paprasčiausiai paudauginant iš nulio.

Toliau pavaizduoti signalai su išskirtais žemais ir aukštesniais dažniais. Tikimės, jog atitinkamai turėsime išskyrę foninę muziką ir balsą.



Pav. 6 Foninės muzikos dažniai ir amplitudės



Pav. 7 Balso dažniai ir amplitudės

Abiem atvejais dažnius ir amplitudes vėl atkuriame į „wave“ signalą atvirkštiniu FFT metodu ir išsaugome failus .wav formatu (3 .wav failai pateikiami kartu su ataskaita).

3. IŠVADOS

FFT metodas plačiai naudojamas šalinant triukšmus, duomenų kompresijai ar analizei ar net išsvestinių skaičiavimui.

FFT ir DFT metodai gražina tą patį rezultatą, tačiau dažniausiai naudojamas FFT metodas, nes jis atlieka mažiau veiksmų.

Foninę muziką pavyko išskirti gan gerai, tačiau ją pašalinti iš balso įrašo sunkiau, nes daug garsų papuola į tą patį dažnių intervalą. Jeigu nekistų nei foninė muzika, nei balso intonacija tuomet FFT metodas duotu žymiai geresnius rezultatus išskiriant ir atkuriant atskirus signalus.

4. PRIEDAI

4.1 Programinio kodo fragmentai

Foninės muzikos išskyrimas:

```
# background music extraction
spectrum = wave.make_spectrum()   # FFT feature extraction
spectrum.low_pass(840)            # cut off higher frequencies
spectrum.plot()
dsp.decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
spectrum.make_wave().make_audio() # reverse FFT
```

Pav. 8 Dažnių didesnių nei 840Hz pašalinimas

```
def make_spectrum(self, full=False):
    """Computes the spectrum using FFT.

    full: boolean, whethere to compute a full FFT
          (as opposed to a real FFT)

    returns: Spectrum
    """
    n = len(self.ys)
    d = 1 / self.framerate

    if full:
        hs = np.fft.fft(self.ys)
        fs = np.fft.fftfreq(n, d)
    else:
        hs = np.fft.rfft(self.ys)
        fs = np.fft.rfftfreq(n, d)

    return Spectrum(hs, fs, self.framerate, full)
```

Pav. 9 „Think DSP“ bibliotekos FFT metodo realizacija

Čia „self.ys“ yra numpy masyvas saugantis reikšmes nuskaitytas iš wav failo.

4.2 Šaltiniai

Allen B. Downey, Franklin W. Olin College of Engineering, „Think DSP: Digital Signal Processing in Python“, 2012