Univerza *v Ljubljani*
Fakulteta *za matematiko in fiziko*

# Matematično-fizikalni praktikum

## Tretja naloga: *Avtokorelacija in FFT*

Simon Bukovšek, 28211067

Mainz, 8. november 2023

**Profesor**: prof. dr. Borut Paul Kerševan

## Naloga: Avtokorelacija in FFT

Na spletni strani MF praktikuma najdeš posnetke oglašanja velike uharice, naše največje sove. Posneti sta dve sovi z minimalnim ozadjem (`bubomono` in `bubo2mono`) in nekaj mešanih signalov, ki zakrivajo njuno oglašanje (`mix`, `mix1`, `mix2` in `mix22`). V signalih `mix2` in `mix22` je oglašanje sove komaj še zaznavno. Izračunaj avtokorelacijsko funkcijo vseh signalov in poskusi ugotoviti, za katero sovo gre pri teh najbolj zašumljenih signalih!

# 1 Uvod

Ta teden nisem imel veliko časa, zato je uvod kar z lista z navodili. Diskretno Fourierovo transformacijo smo definirali kot

$$H_k = \sum_{n=0}^{N-1} h_n \exp(2\pi i k n/N), \qquad k = -\tfrac{N}{2}, \ldots, \tfrac{N}{2},$$

oziroma

$$H_k = \sum_{n=0}^{N-1} W_N^{nk} h_n, \qquad W_N = \exp(2\pi i/N).$$

Ta postopek ima očitno časovno zahtevnost $N^2$. Račun pa je mogoče izvesti tudi z bistveno manj operacijami. Osnovni premislek je razcep

$$H_k = H_k^{\text{sod}} + W_N^k H_k^{\text{lih}} \,,$$

kjer smo transformiranko $H$ izrazili s transformacijama njenih sodih in lihih členov, pri čemer je vsota vsake od transformacij zdaj dolžine $N/2$. Gornjo relacijo lahko uporabljamo rekurzivno: če je $N$ enak potenci števila 2, lahko rekurzijo razdrobimo do nizov, ki imajo samo še en člen. Zanj je transformacija identiteta. Za obrat pri eni vrednosti frekvence (pri danem $m$) je potrebno na vsakem koraku rekurzije le eno množenje s potenco $W$, korakov pa je $\log_2 N$. Skupna časovna zahtevnost je torej le še $N \log_2 N$.

Da ne iščemo pripadnikov niza po vsej tabeli, si podatke preuredimo. Lahko je pokazati, da je v prvotni tabeli treba med seboj zamenjati podatke, katerih vrstna števila v binarnem zapisu so obrnjena: v novem redu jemljemo člene kar po vrsti. Tudi potenc $W$ ne izražamo vedno znova s sinusi in kosinusi, pač pa jih računamo z rekurzijo. Tak ali podoben postopek je osnova vseh algoritmov hitre Fourierove transformacije (FFT).

Z neko transformacijo iz družine FFT bomo izračunali korelacijsko funkcijo dveh signalov. Korelacija periodičnih funkcij $g(t)$ in $h(t)$ s periodo $T$ je definirana kot:

$$\phi_{gh}(\tau) = \frac{1}{T} \int_0^T g(t+\tau)\, h(t)\, \mathrm{d}t \,,$$

oziroma diskretno

$$\phi_{gh}(n) = \frac{1}{N} \sum_{k=0}^{N-1} g_{k+n}\, h_k \,.$$

Računamo torej skalarni produkt funkcij, ki sta časovno premaknjeni za $\tau$ oziroma $n$. Če je za določeno vrednost premika ta funkcija višja kot v okolici, potem to pomeni, da sta si funkciji podobni, le da ju je treba premakniti, da se to vidi.

V primeru, da sta funkciji (signala), ki ju primerjamo, enaki, računamo njuno *avtokorelacijsko funkcijo*: ta je mera za to, ali signal ostaja s pretekanjem časa sam sebi podoben. Če je signal slabo koreliran (sam s sabo), korelacija $\phi_{hh}(n)$ relaksira h kvadratu povprečnega signala $\langle h \rangle^2$, kjer je

$$\langle h \rangle = \frac{1}{N} \sum_{k=0}^{N-1} h_k \,.$$

Iz lokalnih maksimumov v avtokorelacijski funkciji sklepamo na periodičnosti, bodisi popolne ali približne. Pri periodičnih signalih je tudi avtokorelacijska funkcija striktno periodična, za stohastične procese pa je značilna eksponentna avtokorelacijska funkcija. še bolj nas zanima, kako *hitro* se korelacija izgublja: računamo rajši reskalirano obliko avtokorelacije

$$\widetilde{\phi}_{hh}(n) = \frac{\phi_{hh}(n) - \langle h \rangle^2}{\phi_{hh}(0) - \langle h \rangle^2} \,,$$

kjer je imenovalec nekakšno merilo za varianco signala,

$$\sigma^2 = \phi_{hh}(0) - \langle h \rangle^2 = \frac{1}{N} \sum_{k=0}^{N-1} \left(h_k - \langle h \rangle\right)^2 \,.$$

Pri zgornjih enačbah moramo še "peš" poskrbeti za periodično zaključenost signala pri $n = N$, torej da je perioda enaka velikosti vzorca. Če tega ne moremo narediti, je bolj pravilna definicija avtokorelacije

$$\phi_{hh}(n) = \frac{1}{N-n} \sum_{k=0}^{N-n-1} h_{k+n}\, h_k \,.$$

Praktičen račun po zgornji formuli lahko postane za velike vzorce prezamuden. Avtokorelacijo rajši računamo s FFT (DFT) $\mathcal{F}$, saj je korelacija obratna Fourierova transformacija $\mathcal{F}^{-1}$ produkta Fourierovih transformacij $\mathcal{F}$, torej z $G = \mathcal{F}g$ in $H = \mathcal{F}h$ dobimo

$$\phi_{gh}(n) = \frac{1}{N-n} \mathcal{F}^{-1} \left[ G \cdot (H)^* \right]$$

oziroma

$$\phi_{hh}(n) = \frac{1}{N-n} \mathcal{F}^{-1} \left[ \, |H|^2 \, \right] \,.$$

Za račun s FTT signale dolžine $N$ najprej prepišemo v dvakrat daljše, periodično zaključene podatkovne nize, $\widetilde{h}_n = h_n$, $\widetilde{h}_{n+N} = 0$ za $n = 0, \ldots, N-1$ in $\widetilde{h}_{n+2N} = \widetilde{h}_n$. Tedaj se avtokorelacija zapiše v obliki

$$\phi_{hh}(n) = \frac{1}{N-n} \sum_{k=0}^{2N-1} \widetilde{h}_{k+n}\, \widetilde{h}_k \,,$$

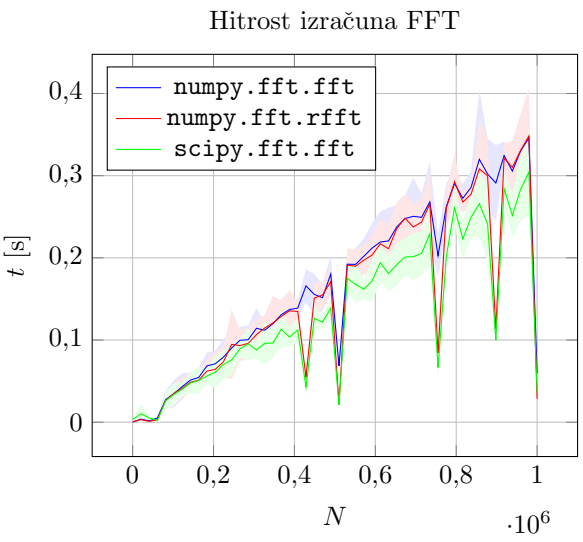
kar lahko izračunamo s FFT.

# 2 Rezultati

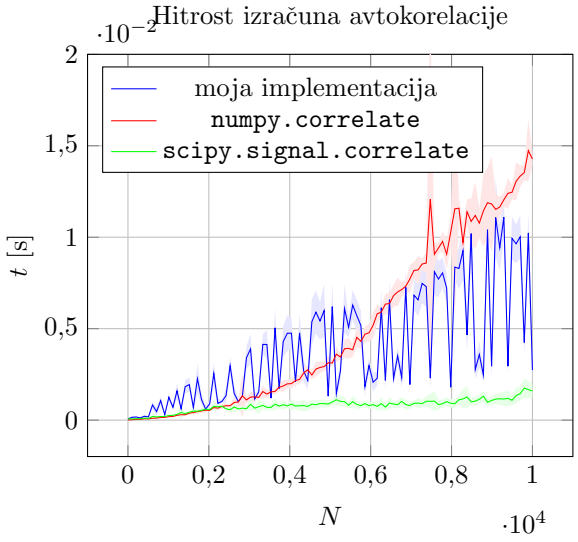
## 2.1 Hitra Fourierova transformacija

Preden se lotimo dejanskega računanja, si poglejmo časovne zahtevnosti metod, ki so nam na voljo za uporabo. Kot priporočeno v navodilih, je najbolje uporabiti FFT, vendar imamo za to več možnosti. Knjižnica `numpy.fft` nam ponuja dve metodi: `fft` in `rfft`. Slednja deluje le na realnih signalih in pri tem izkorišča določene simetrije v postopku. Vprašanje je, seveda, koliko se to res pozna. Še ena možnost je implementacija FFT iz knjižnice `scipy.fft`. Prikaz časovnih zahtevnosti je na Sliki 1a. Podatki so bili zbrani s transformacijo seznama naključnih števil dolžine $N$ s porazdelitve $\mathcal{U}_{[0,1]}$. Za vsako izmed testiranih števil je bila transformacija ponovljena desetkrat za namene statistične analize. S polno črto je prikazano povprečje desetih izračunov različnih naključnih seznamov iste dolžine, s svetlo barvo pa razpršenost do ene standardne deviacije. Opažanja so zbrana na naslednjem seznamu:

- Metoda `numpy.fft.fft` ni bistveno slabša od `fft.fft.rfft`, razen v nekaterih točkah, ko `rfft` bistveno odstopi od približno linearnega trenda.
- Metoda `scipy.fft.fft` je v povprečju za 5 % do 10 % hitrejša od `numpy.fft.fft`. Razlika je konsistentna, vendar bi lahko na drugem računalniku prišlo do drugačnih rezultatov. Zaradi majhnih razlik bom vseeno uporabljal `numpy.fft.fft`.
- Časovna zahtevnost je dokaj linearna, kar je podobno teoretični napovedi $n \log n$. Pri tako velikih $n$ se logaritemski faktor na grafu skorajda ne pozna več.
- Pri določenih dolžinah vhodnega signala se predvsem pri `scipy`-jevi implementaciji čas znatno zmanjša. To so verjetno tiste vrednosti, ki so blizu številom z veliko potenco pri dvojki v praštevilskem razcepu. Na primer, za $N = 1\,000\,000$ je program v povprečju porabil desetkrat manj časa, kot bi predvideval trend. Število milijon je seveda blizu $2^{20} = 1\,048\,576$. Učinkovitost pri potencah števila 2 pa je posledica pristopa "deli in vladaj" v implementacijah algoritmov.




**(a)** Primerjave časovnih zahtevnosti različnih implementacij FFT do dolžine vhodnega signala $N = 10^6$, kar je dosti več, kot potrebujem za nalogo. V svetlem je prikazan en standardni odklon.




**(b)** Časovna zahtevnost različnih izračunov korelacijske funkcije. V svetlem je prikazana razpršenost rezultatov do enega standardnega odklona.

**Slika 1:** Primerjave časovne zahtevnosti različnih potencialno uporabnih metod.

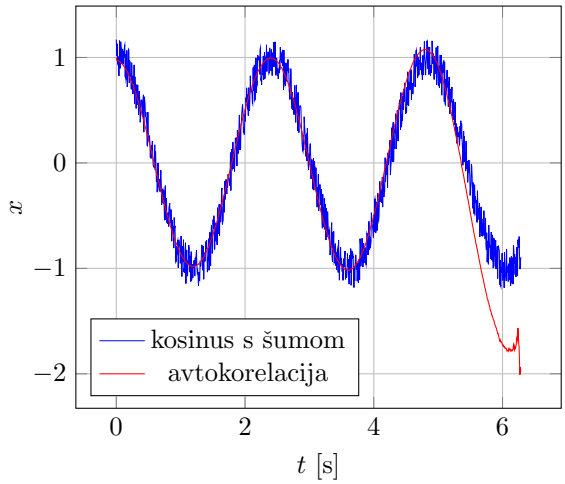
## 2.2 Korelacijska funkcija

Na podlagi analize iz prejšnjega podpoglavja sem za vse nadaljnje programje uporabil metodo `numpy.fft.fft`, saj med vsemi metodami res ni bistvene razlike. Naslednji korak je bil sestaviti korelacijsko funkcijo. Tudi za ta primer imata `numpy` in `scipy` že pripravljene rešitve, zato me je zanimalo, kako se moja implementacija primerja z njima. Rezultati so prikazani na Sliki 1b. Podatki so bili zbrani na enak način kot v prejšnjem podpoglavju. Opaziti je izrazit kvadratičen trend pri metodi `numpy.correlate`, kar je v skladu z opozorilom v dokumentaciji te funkcije. Korelacijo namreč računa po definiciji brez FFT. Na drugi strani je `scipy.signal.correlate` v primerjavi znatno hitrejši in po grobi oceni raste približno linearno. Ta metoda uporablja FFT, zato sem upal, da bo moja implementacija bližje zeleni kot rdeči črti. Po končani meritvi lahko rečem, da se preprosto ne more odločiti, kateri od ostalih dveh bi sledila. Pri nizkih dolžinah signalov je seveda slabša od obeh ostalih, saj je (kot vedno) Python zelo počasen v primerjavi z jeziki, v katerih se izvajajo metode iz `numpy` in `scipy` knjižnic. Kljub precejšnji stohastičnosti v kontekstu časa izvajanja pa je približno jasen linearen (ali pa vsaj sub-kvadraten) trend, s katerim prehiti `numpy`-jevo metodo pri dolžini $N = 6000$. Kljub precejšnji počasnosti sem v nadaljevanju uporabljal svoj program, saj bi bila v nasprotnem primeru potrata časa, ki sem ga vložil v implementacijo.

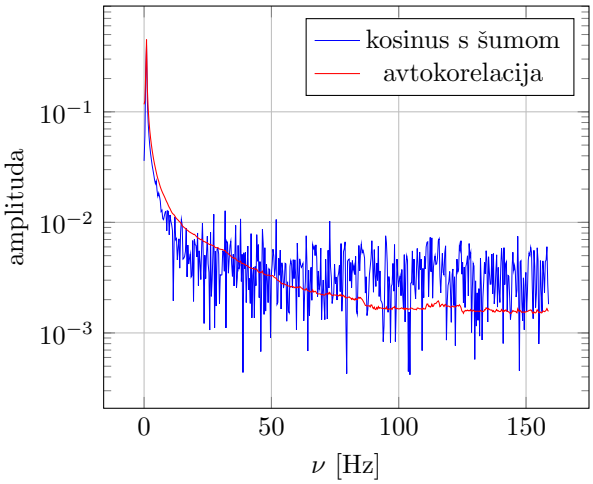
## 2.3 Testiranje avtokorelacije

Poglejmo si, če smo program sploh pravilno napisali. To je najlažje preveriti na primeru preprostega sinusnega vala z nekaj šuma. Na Sliki 2a je prikazana funkcija $f(x) = \cos(2{,}6x) + 0{,}2\hat{x}$, kjer je šum $\hat{x}$ žreban iz enakomerne porazdelitve na $[-1, 1]$. Opazimo, da je na avtotkorelirani funkciji bistveno manj šuma, kar je natanko cilj te metode. Na koncu rdečega grafa je videti nekaj odstopanj in več šuma, za kar je krivo znatno skaliranje funkcije (spomnimo se faktorja $(N-n)^{-1}$ pred vsoto v avtokorelaciji). Omeniti je potrebno še, da bi zelo podobno avtokorelacijo dobili, tudi če bi imeli za začetno funkcijo sinus. V tem primeru bi avtokorelacija še vedno imela obliko kosinusa, saj mora imeti pri ničli vrh. Na sliki 2b pa je prikazana Fourierova transformacija obeh funkcij, na kateri se tudi lepo vidi, kako avtokorelacija izbriše šum oziroma vse neželene frekvence.

Drug primer je bistveno bolj zašumljen in za razliko od prejšnjega uporablja beli šum, ki je mnogo bolj prisoten v naravi. Tokrat je funkcija $f(x) = \cos(3{,}1x) + \hat{x}$, kjer je šum $\hat{x}$ porazdeljen po normalni porazdelitvi $\mathcal{N}(0, 1)$. To si lahko interpretiramo tudi kot enakomerno prisotnost vseh frekvenc do Nyqvistove. Rezultati so prikazani na Sliki 3. Opaziti je, da je amplituda avtokorelirane funkcije nižja kot v prejšnjem



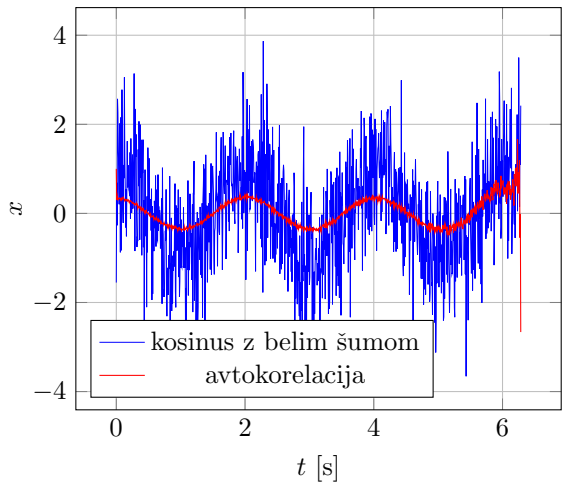

**(a)** Testna funkcija $\cos(2{,}6x) + 0{,}2\hat{x}$, kjer je $\hat{x} \sim \mathcal{U}_{[-1,1]}$, in njena normirana avtokorelacijska funkcija.



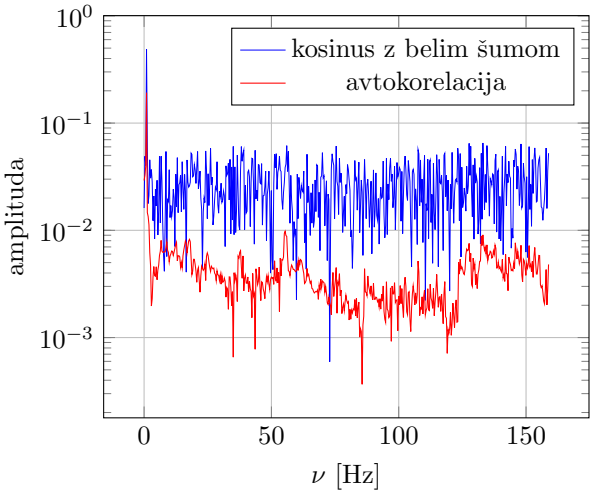

**(b)** Fourierova transformacija funkcije z leve slike in njene normirane avtokorelacijske funkcije.

**Slika 2:** Primer avtokorelacije z "uniformnim" šumom.

primeru, kar si lahko interpretiramo kot misel, da je avtokorelacija manj prepričana v svoje rezultate. Še vedno pa smo lahko izolirali periodičnost od večine šuma. Na Sliki 3b sta prikazani še Fourierovi transformaciji. Originalna funkcija ima šum prisoten po vseh frekvencah skoraj enakomerno, avtokorelacija pa to odpravi.




**(a)** Testna funkcija $\cos(3{,}1x) + \hat{x}$, kjer je $\hat{x} \sim \mathcal{N}(0,1)$, in njena normirana avtokorelacijska funkcija.



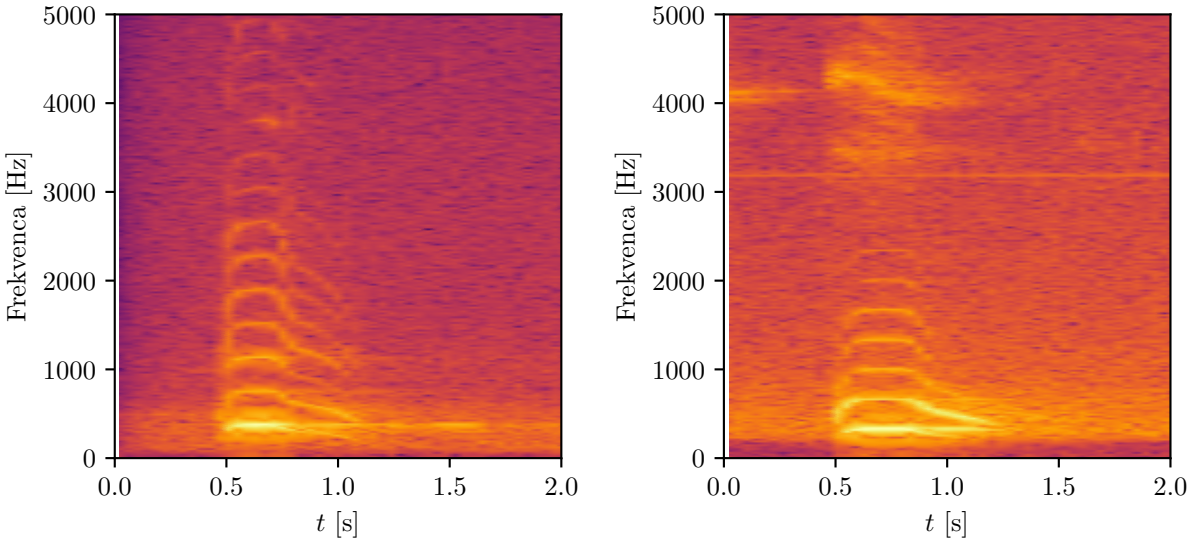

**(b)** Fourierova transformacija funkcije z leve slike in njene normirane avtokorelacijske funkcije.

**Slika 3:** Primer avtokorelacije z "belim" (normalnim Gaussovim) šumom.

## 2.4 Sove

Na spletni učilnici smo dobili šest posnetkov sov. Prva dva sta jasna posnetka dveh sov, ki ju bomo imenovali Sova 1 in Sova 2. Za vizualno predstavo bi lahko naredili Fourierovo analizo celotnega posnetka (in to bomo naredili), vendar je za neperiodične zvoke bolj naravno, da se jih predstavi s pomočjo *spektrograma*. To je dvo-dimenzionalen graf, ki na eni osi prikazuje čas, na drugi frekvenco, barva pa predstavlja amplitudo frekvence ob tistem času. Spektrograma obeh sov sta prikazana na Sliki 4. Pri izrisovanju spektrograma gre za ravnotežje med ločljivostjo v času in ločljivostjo v frekvenci. Če se eno poveča, se

drugo nujni zmanjša (kot pri Heisenbergovem načelu, saj je v ozadju ista matematika). Za graf sem izbral časovno ločljivost $t_0 = 46\,\text{ms}$, kar da frekvenčno ločljivost $\nu_0 = 22\,\text{Hz}$. Ker ljudje zaznavamo zvok logaritemsko, je tudi smiselno prikazovati logaritme amplitud. Oba grafa opišeta veliko stvari, ki bi jih lahko povedali ob poslušanju posnetkov, hkrati pa razkrijeta še več. Lahko razberemo, da je na posnetku sove 1 manj šuma kot pri posnetku sove 2, prav tako pa približen potek skovika. Opazimo tudi višje harmonike osnovne frekvence. Na Sliki 4b pa je opazen še en vir pri približno 3,1 kHz.




**(a)** Sova 1 **(b)** Sova 2

**Slika 4:** Spektrograma obeh sov. Prikazani so logaritmi amplitud. Časovna ločljivost je 46 ms, kar da frekvenčno ločljivost 22 Hz.

Da bi sove lahko bolj natančno primerjali, pa potrebujemo bolj natančen prikaz Fourierove transformacije. Za obe sovi hkrati in njuni avtokorelaciji je transformacija na Grafu 5a. Avtokorelaciji očitno zmanjšata šum, hkrati pa lahko tudi ločimo obe sovi glede na višino tona (to sicer lahko slišimo tudi na posnetku, ampak graf je bolj kvantitativen način prikaza).

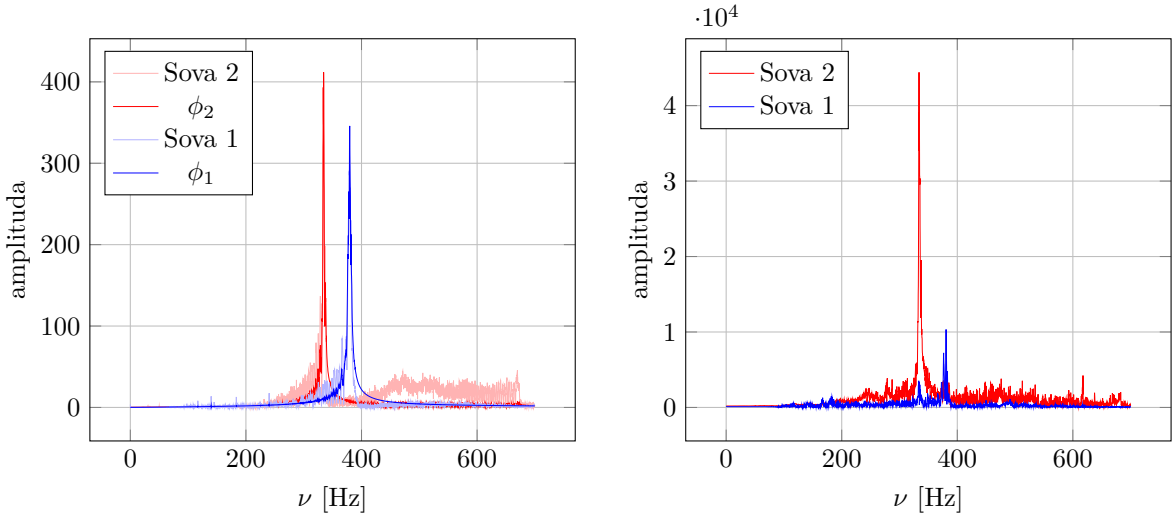
Naredil sem avtokorelacije vseh šestih posnetkov, ampak je grafov toliko, da sem jih raje vključil v dodatek. S Slike 7 lahko vidimo, da avtokorelacija v splošnem dobro deluje. Na četrti in peti sliki se lepo vidijo črički na frekvencah okoli 2400 Hz in 4800 Hz, kar je točno eno oktavo narazen.

## 2.5 Korelacija sov

Tokrat smo imeli že vnaprej podane podatke o tem, katera sova je na katerem posnetku. Predstavljajmo si, da recimo za zadnji posnetek tega ne bi vedeli. En način, kako bi to lahko določili je s pomočjo korelacijske funkcije. Izračunal sem korelacijo med posnetkom prve sove in posnetkom `mix22.txt` ter enako za drugo sovo. Nato sem izračunal Fourierovi transformaciji obeh korelacij in jih narisal na Graf 5b. Kot kaže, je vrh pri Sovi 2 precej višji kot pri Sovi 1, zato lahko s precejšno gotovostjo trdimo, da je na posnetku Sova 2. Drži sicer, da je posnetek Sove 2 nekoliko glasnejši od Sove 1 in bi že zaradi tega lahko prišlo do različno visokih vrhov, vendar je razlka v jakosti občutno manjša od razlike v višini vrhov.

## 2.6 Dodatna naloga

Avtokorelacija se ne uporablja samo za odpravljanje šuma na periodičnih signalih, ampak lahko presenetljivo deluje tudi na kakšnih bolj nepričakovanih stvareh, recimo borznem tečaju ali analizi slik. Za dodatno nalogo sem se odločil narediti nekaj bistveno manj uporabnega, ampak ker smo imeli proste roke, sem si mislil, da vse, kar ni prepovedano, je dovoljeno. Želel sem neka unikatnega, in ker to poročilo pišem iz Mainza, sem pomislil, da lahko vključim kaj lokalnega: najbrž ni še nihče pred mano poskusil narediti avtokorelacije pretoka Rena v Mainzu. Rezultat je na Sliki 6. Na žalost sem uspel dobiti le podatke za

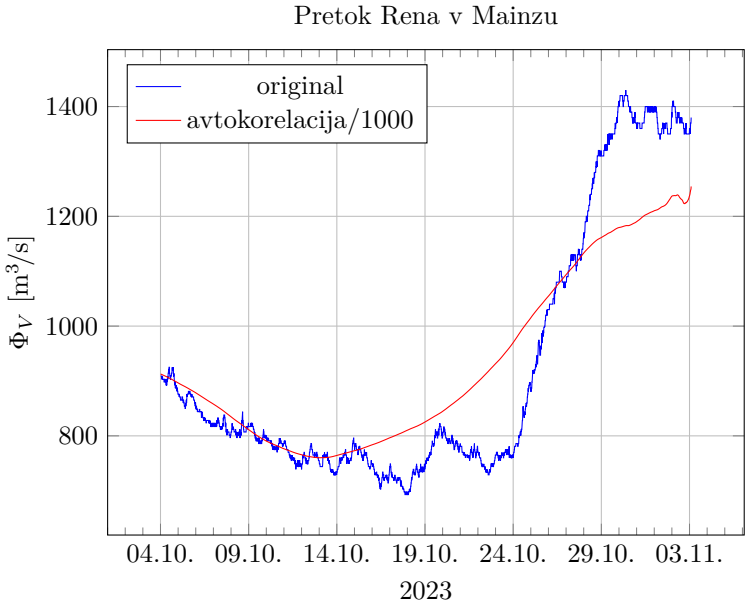



**(a)** Fourierova transformacija posnetkov sov in njunih avtokorelacij. Z rdečo je označena Sova 1, z modro pa Sova 2. Originalni posnetki so v svetlem, avtokorelacije pa v temnem odtenku.

**(b)** Fourierova transformacija posnetka `mix22.txt`, koreliranega s posnetkoma obeh sov. Z rdečo je označena Sova 1, z modro pa Sova 2.

**Slika 5:** Analiza sovjih oglašanj.

prejšnji mesec. Pričakovano rezultat ni pretirano navdihujoč. Korelacijo sem skaliral in prestavil, tako da se je približno ujela z grafom pretoka. Šum je definitivno izbrisan, vendar se zdi, kot da bi nastavitve glajenja nastavil na maksimum. Zaradi neperiodičnosti težko pričakujemo kaj boljši rezultat.




**Slika 6:** Avtokorelacija pretoka Rena v Mainzu.

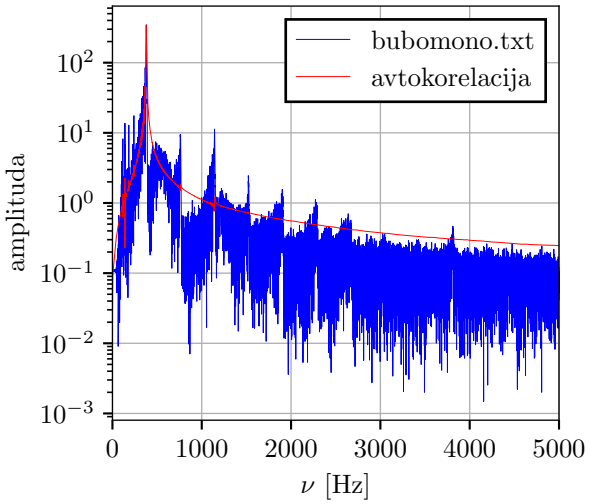
# 3 Zaključek

Naloga nam je predstavila analizo signalov s pomočjo Fourierove transformacije in (avto)korelacije. V tokratnem zaključku bi rad delil dve izkušnji. Prva se je zgodila, ko sem prvič videl rezultate hitrosti

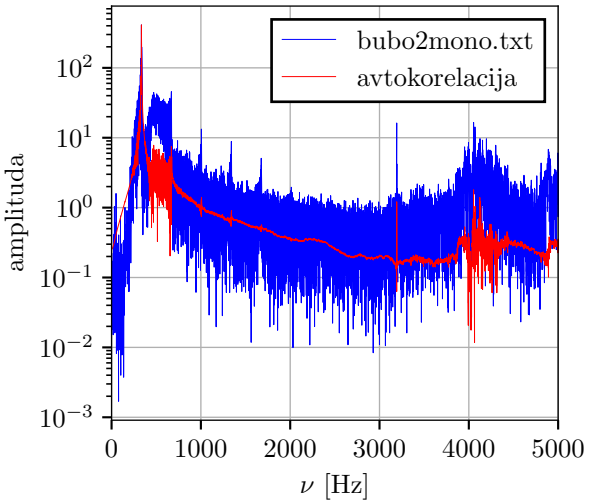
korelacijskih algoritmov. Zelo me je presenetilo, da je `numpy`-jev algoritem tako počasen. Seveda sem se potem počutil toliko bolj neumnega, ko sem se odpravil brati dokumentacijo (ki bi jo, iskreno, moral prebrati že prej) in prebral na veliko opisano opozorilo v smislu "Ne uporabljajte te funkcije za velike vzorce, raje uporabite `scipy`!" Drugo presenečenje se je zgodilo, ko sem bistveno preveč časa porabil za izračun frekvenčne ločljivosti $\nu_0$ spektrograma, če je časovna ločljivost $t_0$. V računu sem približno štirikrat uporabil vzorčno frekvenco 44,1 kHz in se spraševal, ali nisem po nesreči tiste dvojke pri Nyqvistovi frekvenci uporabil dvakrat, nakar sem prišel do rezultata $t_0\nu_0 = 1$. Še zdaj nisem prepričan, ali je bil ta rezultat že ves čas popolnoma očiten. Naj bralec oceni...

# Dodatek

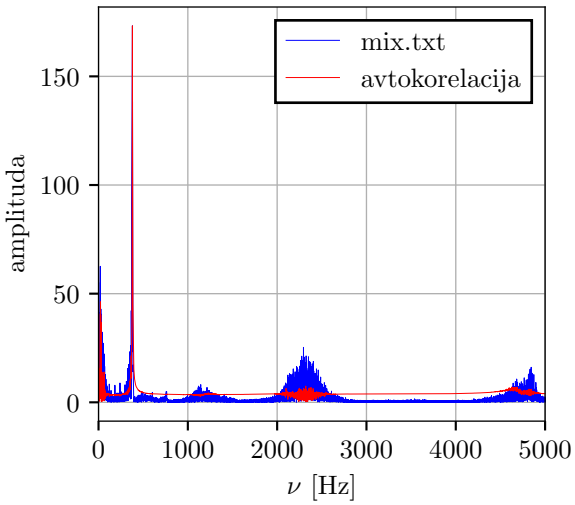
Na naslednji strani najdete Fourierove transformacije posnetkov in njihovih korelacij.



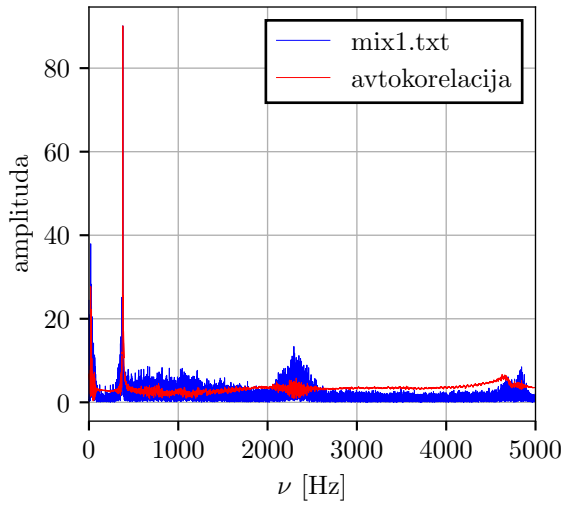

**(a)** Sova 1.



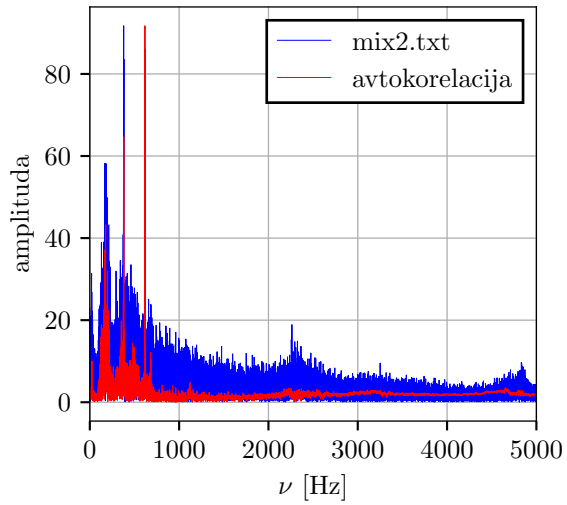

**(b)** Sova 2.



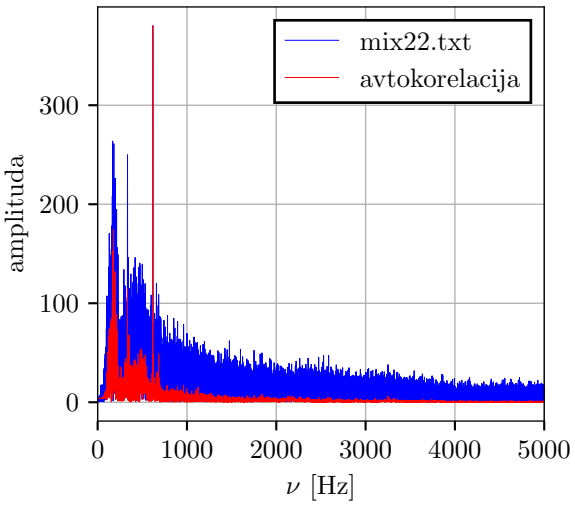

**(c)** Sova 1 s črički.




**(d)** Sova 1 s črički ob potoku.




**(e)** Sova 1 s črički ob deroči reki.




**(f)** Sova 2 ob deroči reki.

**Slika 7:** Fourierove transformacije posnetkov in njihovih korelacij.