

5. naloga: Hitra Fourierova transformacija in korelacijske funkcije

Samo Krejan, 28231092

Diskretno Fourierovo transformacijo smo definirali kot

$$H_k = \sum_{n=0}^{N-1} h_n \exp(2\pi i kn/N), \quad k = -\frac{N}{2}, \dots, \frac{N}{2},$$

ozziroma

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

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 sinusom in kosinusom, 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:

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

ozziroma 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 τ ozziroma n . Če je za določeno vrednost premika ta funkcija višja kot v okolini, potem to pomeni, da sta si funkciji podobni, le da ju je treba premakniti, da se to vidi.

V primeru, da sta funkciji (signal), 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 maksimov 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

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

kjer je imenovalec nekakšno merilo za varianco signala,

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

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

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

oziroma

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

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

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

kar lahko izračunamo s FFT.

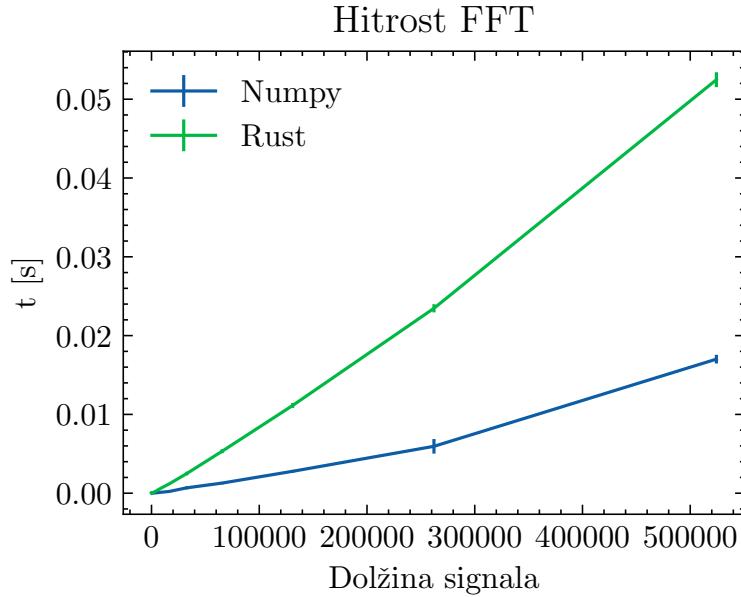
1 Naloga

Na spletni strani MF praktikuma najdeš posnetke oglašanja velike uharice, naše največje sove. Posneti sta dve sovi z minimalnim ozadjem (`bubomon` in `bubo2mon`) 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!

Poglejte si rutine `four1` iz Numerical Recipes ali knjižnice `fftw3`, ki je še dosti hitrejša. V okolju Python so te rutine vključene v ’fft’ paket. (Pri tako velikih vzorcih je skorajda nujno uporabiti FFT namesto počasne navadne DFT.)

1.1 FFT

Nalogo sem začel tako, da sem najprej naredil lastno implementacijo hitre Fourierove transformacijev Rust-u. Ta mi sicer ni uspela najbolje saj je med drugim delovala zgolj za sezname katerih dolžina je potenca števila 2 in pa, čeprav je dosegala natančnosti v bližini numerične zmogljivosti dvojne natančnosti, je bila mnogo počasnejša od `numpy.fft.fft` implementacije (slika 1), tako da sem se v nadaljevanju naloge odločil uporabljati kar `numpy` implementacijo.



Slika 1: Čas potreben za izvedbo FFT v odvisnosti od velikosti seznama (potence števila 2). Kvalitativno lahko vidimo, da časovna zahtevnost res ustreza $O(n \log n)$ in pa tudi, da je numpy implementacija mnogo boljša.

1.2 Avtokorelacijska funkcija

Ker bomo kasneje naredili avtokorelacijo signalov z hitrim vzorčenjem je pomembno, da poskrbimo, da je algoritem s katerim računamo avtokorelacijo čim bolj učinkovit. Zato sem analiziral tri implementacije avtokorelacije, sicer:

1. Osnovna implementacija v Pythonu po definiciji iz uvoda:

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

2. Po definiciji iz uvoda z žero paddingža uporabo *numpy*:

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

3. Navsezadnje pa še z uporabo FFT algoritma, kot je opisano v uvodu:

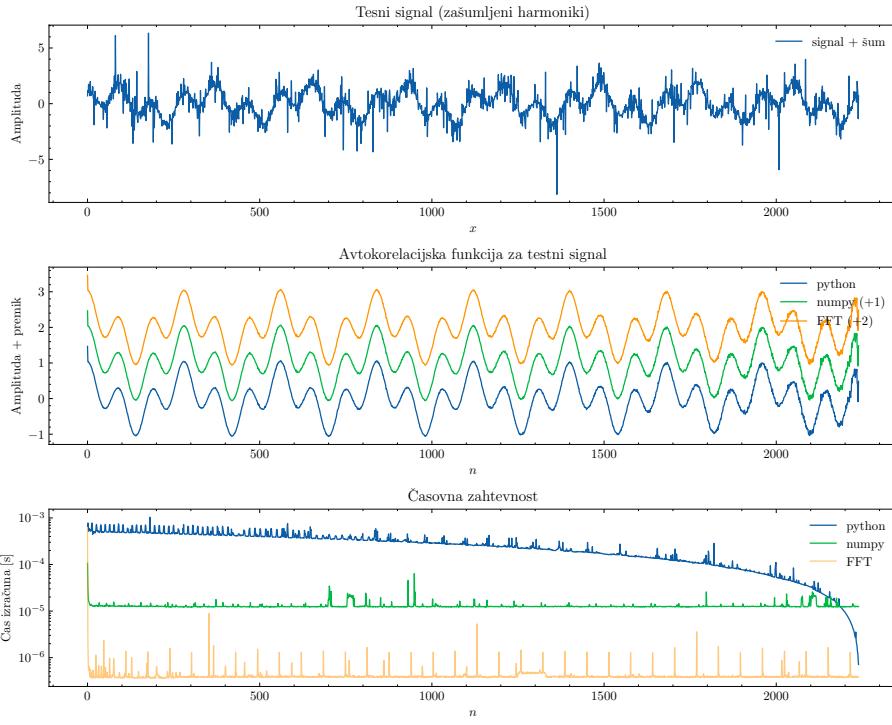
$$\phi_{hh}(n) = \frac{1}{N-n} \mathcal{F}^{-1} [|H|^2] .$$

Uspešnost avtokorelacije zašumljenega signala harmonikov in potrebnim časom za vsako iteracijo, sem prikazal na grafu 2

Takoj opazimo, da je implementacija z FFT daleč najhitrejša, zato sem od te točke naprej uporabljal kar to. Poleg tega opazimo tudi, da je avtokorelirana funkcija izgubila precej šuma, kar je odlična posledica avtokorelacije, ki nam bo kasneje prišla prav.

1.3 Analiza posnetkov

Na tej točki sem končno opremljen z vsemi potrebnimi orodji, da se lotim analize posnetkov sov. Najprej sem uporabil najbolj primitivno izmed orodij - lastna ušesa in iz poslušanja sklepal, da posnetek mix22 pripada sovi bubo2, medtem ko se mi zdi, da ostali pripadajo sovi bubo.



Slika 2: Signal, njegova avtokorelacija (bralec naj bo pozoren na zamike za namen lepšega prikaza), ter časovna zahtevnost

Predno pa sem se lotil potrjevati zgornjo hipotezo, sem želel dobiti še nekaj več občutka za posamezne posnetke, tako da sem z *matplotlib* funkcijo *specgram* najprej narisal spektrogram posameznih posnetkov. Na ta način sem dobil grafe na sliki 3

Sovi bo verjetno najlažje ločiti na podlagi frekvence skovikov, zato sem na prvi točki naredil zgolj Fourierovo transformacijo in dobil graf 4

Hitro vidimo nekaj ujemanja med vrhovi frekvenc, a si sam, zaradi očitno mnogo šuma ne upam trditi, da dobljeni rezultati potrdijo mojo hipotezo. Zato sem se odločil narediti še relativno avtokorelacijsko funkcijo vseh posnetkov. Na ta način sem dobil graf 5

Z dobljenimi grafi si nisem znal pomagati, nato pa sem se spomnil, kako avtokorelacija "polepša" podatke in bo tako Fourierova transformacija na avtokorelacijski funkciji izgledala bolje. Tako sem se odločil izvesti najprej avtokorelacijo in nato FFT in pa tudi dvakrat avtokorelacijo in nato FFT. Na ta način sem dobil slike 6 in 7

Predvsem na zadnji sliki sedaj res lepo vidimo, da imata signala frekvenco pri precej distinktnih frekvencah (različni frekvenci skovika), tako da lahko končno z odločnostjo trdim, da imam očitno dovolj dober posluh, da lahko ločim dve sovi. Avtokorelacija in pa FFT sta se seveda pri tem izkazala kot dva ključna algoritma, ki misim, da sem jih uporabil uspešno.

2 Dodatna naloga

Za dodatno nalogo sem se odločil na hitro analizirati uspešnost generatorjev naključnih števil. Primerjal sem numpy-jev random.random algoritmom, in pa starodaven RANDU algoritmom, ki ga je v 60ih razvil IBM.

Naloge sem se lotil tako, da sem generiral seznamu naključnih števil dolžine 2^{25} najprej z enim in nato še z drugim algoritmom, in nato izvedel avtokorelačijsko funkcijo na posameznem seznamu. Idealen generator naključnih števil bi imel pri zelo dolgem seznamu avtokorelačijsko funkcijo povsod enako nič, razen pri zamiku nič. Kaj sem dobil dejansko za omenjena primera je razvidno na slikah 8 in 9. Kot vidimo je numpy-jev generator precej bolj uspešen, saj pri RANDU opazimo nekaj koreliranosti. Kljub temu mislim, da je glede na starost generatorja tudi RANDU precej uspešen.

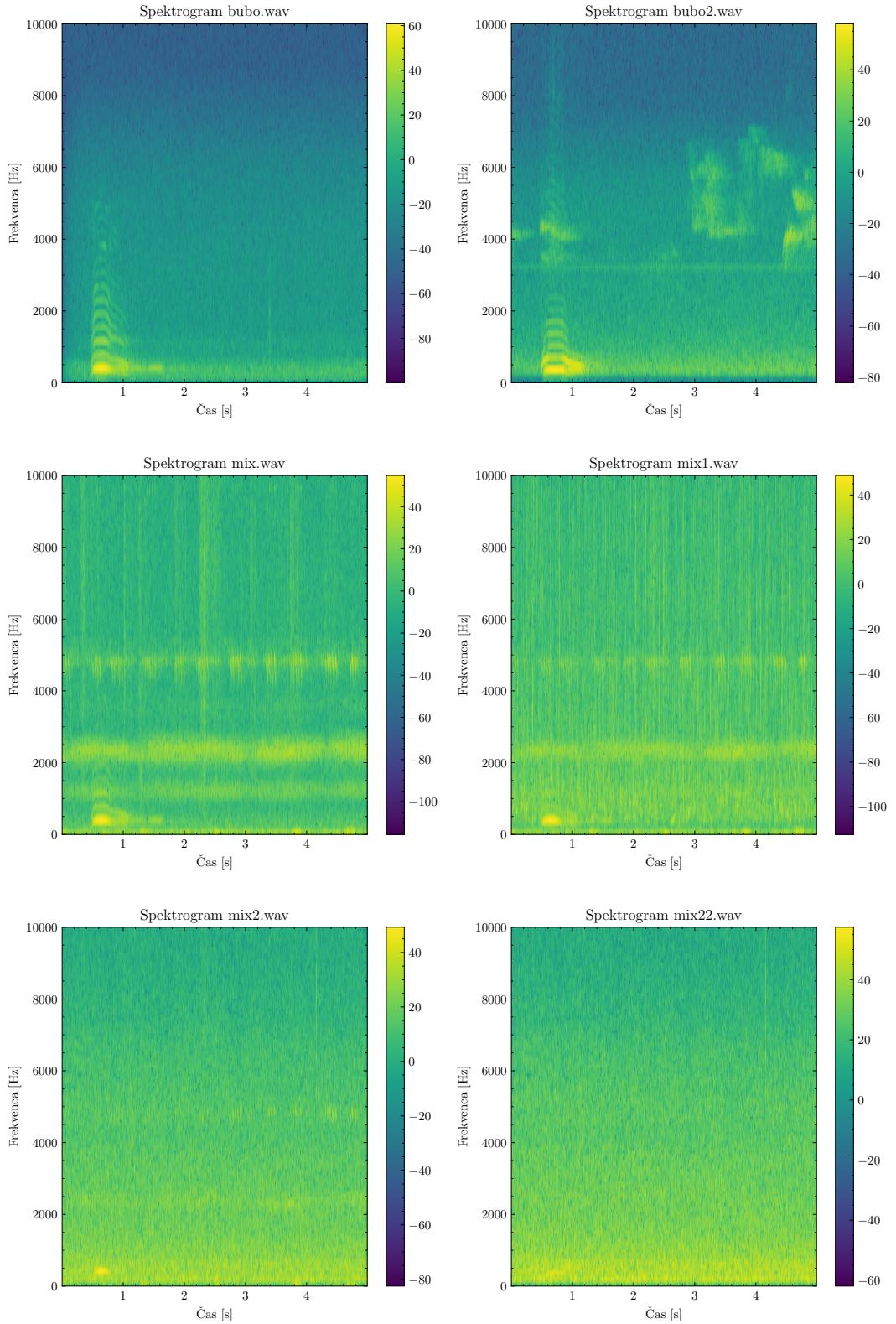
3 Zaključek

Vaja je bila gotovo uspešeno izvedena, saj smo dosegli cilj in uspeli identificirati sove, tudi ko je bilo v ozadju mnogo ostalih zvokov. To nam je uspelo s pomočjo avtokorelacije in FFT algoritma, ki se izkažeta, kot dva izmed bolj pomembnih algoritmov, ko prido do iskanja vzorcev v signalih.

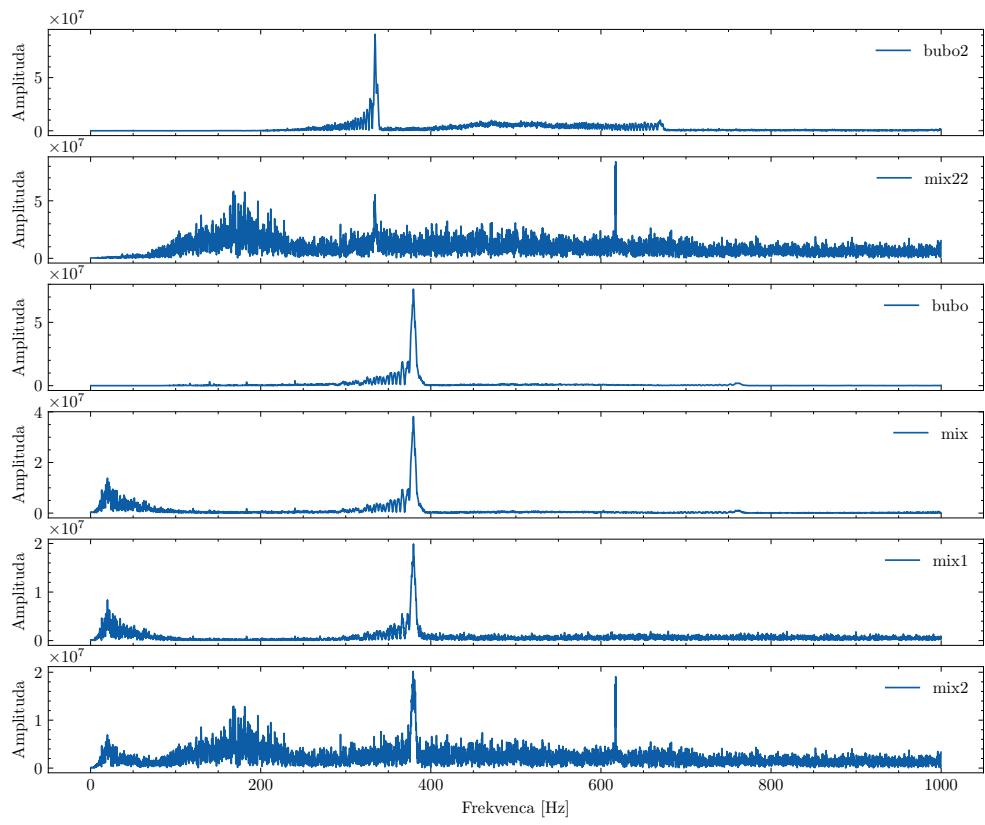
Kljub eleganci algoritmov, mi je bila ta vaja najmanj všeč med vsemi do sedanjimi, verjetno tudi zato, ker sem imel zanjo najmanj časa in sem se tako težje poglobil v naloge. Upam, da se to ne odraža nikjer drugje kot v uri oddaje naloge.

Sigurno se pomankanje časa odraža v poročilu pri poziciji grafov, ki so po dobrini uriti srdite borbe ostali nepremagani in samozavestno na koncu poročila. Upam, da to ne moti preveč.

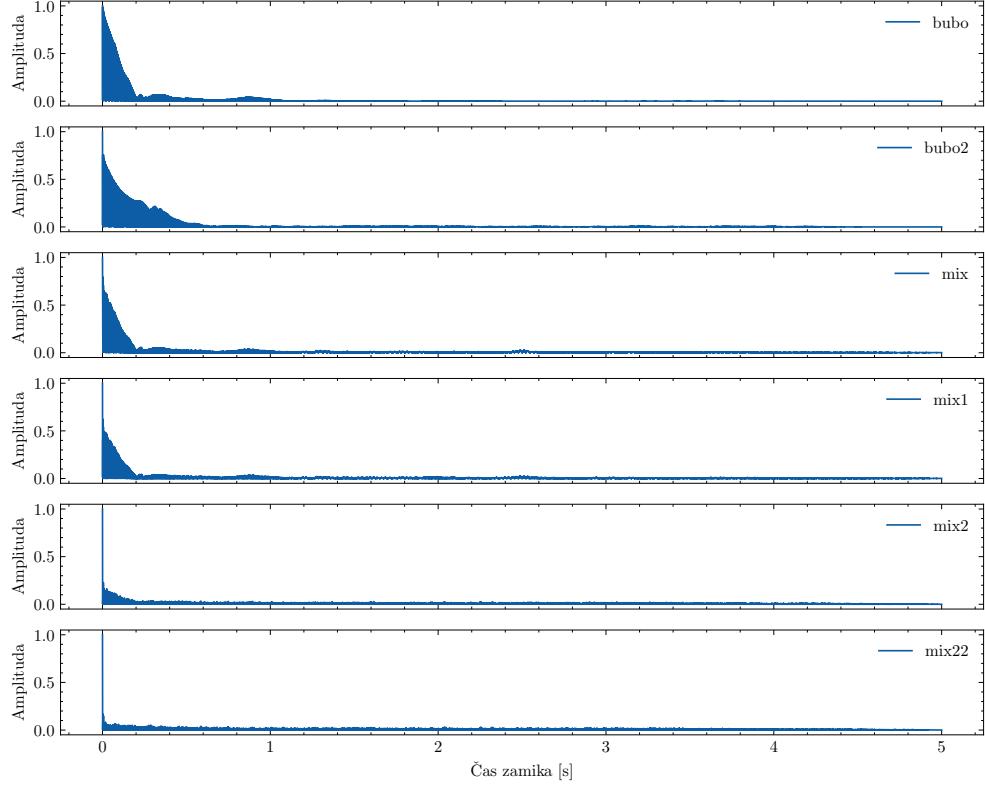
Na koncu, sem se kljub veliko preglavicam naučil mnogo novega in morda bom po tem, ko se naspim, priznal, da je bilo celo vredno truda :)



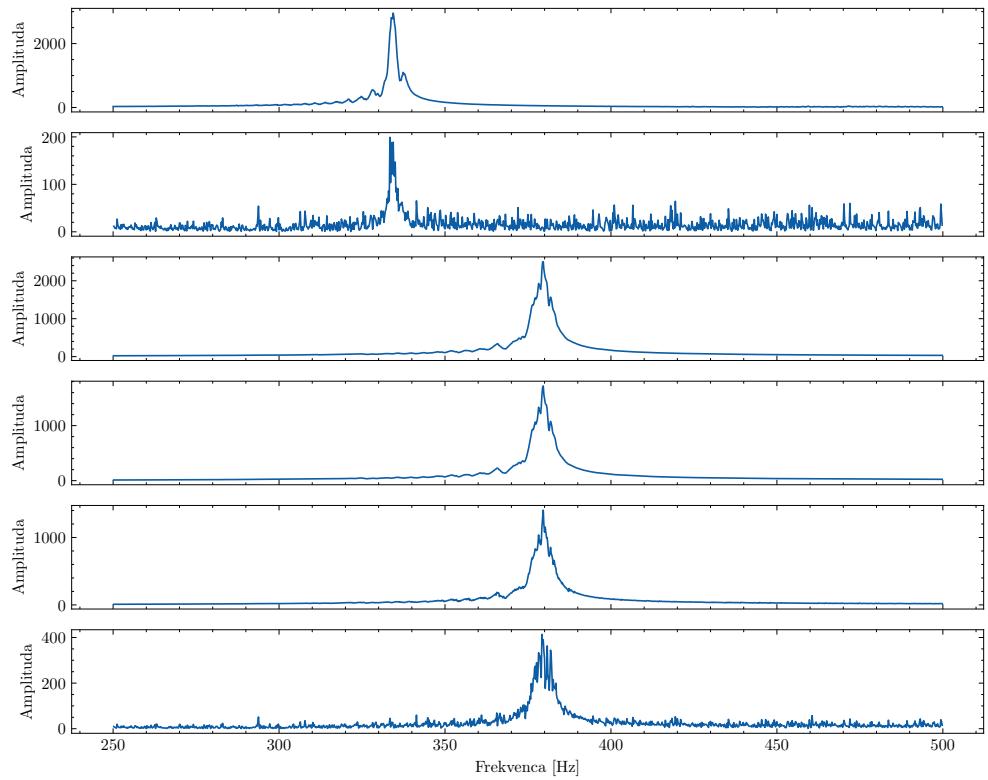
Slika 3: Spektogrami posnetkov. Nekatere lastnosti posnetkov lahko razberemo že iz spektrograma. Vidimo lahko naprimer, kako močno sta zašumljena posnetka mix2 in mix22, saj se na njima sam skovik vidi le še z nekaj domišljije. Poleg tega lahko na spektrogramu posnetka bubo vidimo celo višje frekvence skovika



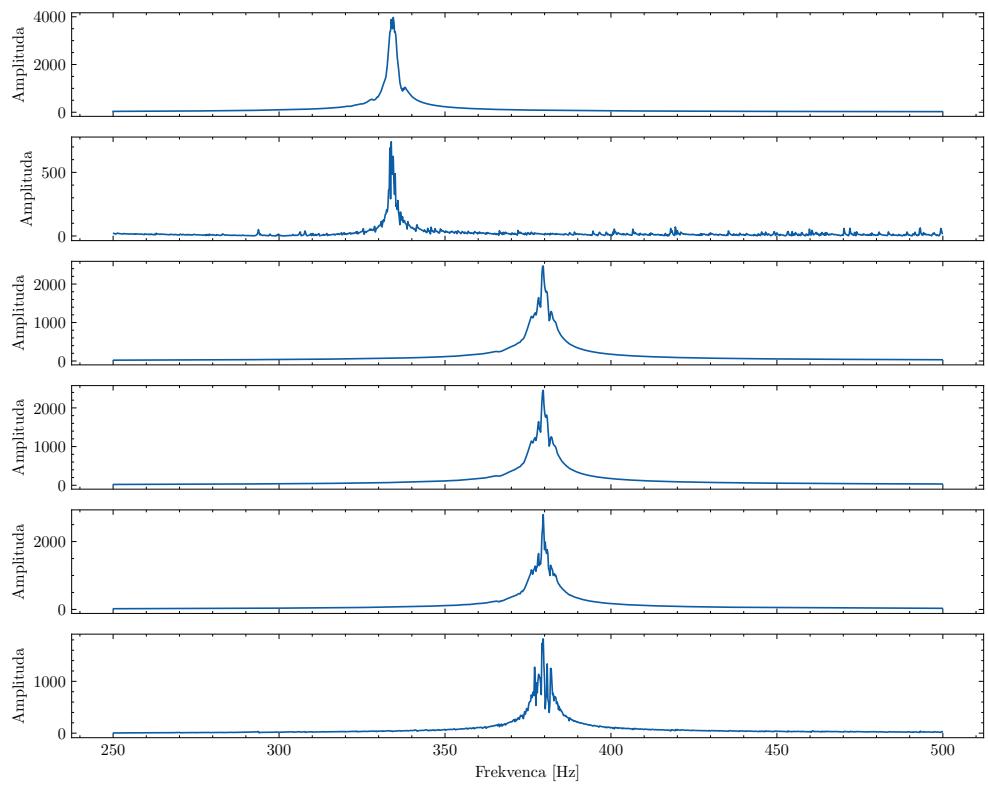
Slika 4: Fourierova transformacija vseh posnetkov



Slika 5: Avtokorelacijska funkcija vseh posnetkov

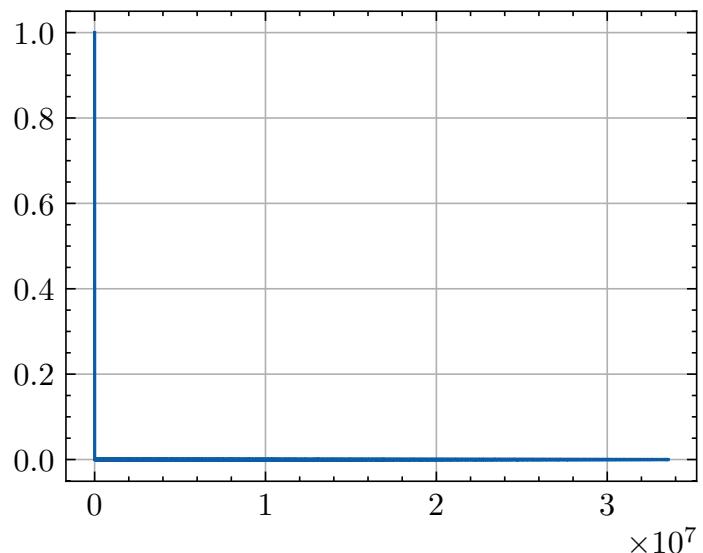


Slika 6: Avtokorelacijska funkcija Fourierovo transformirana



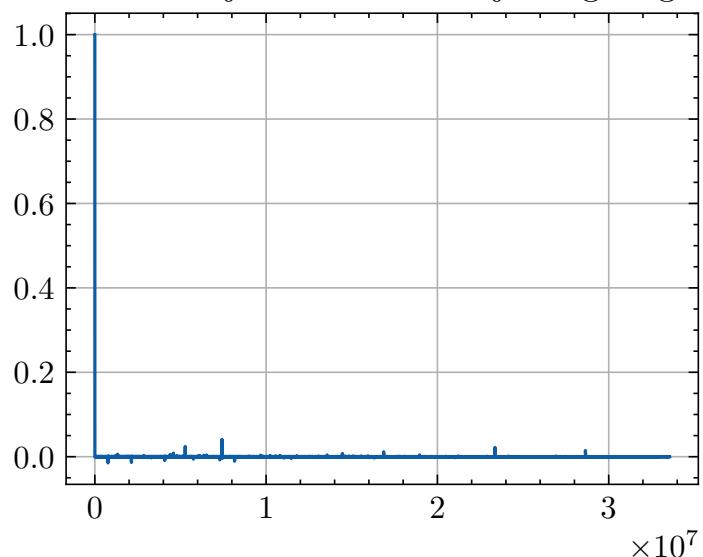
Slika 7: Avtokorelacijska funkcija avtokorelacijske funkcije Fourierovo transformirana

Avtokorelacija NUMPY nakljucnega signala



Slika 8: Avtokorelacijska funkcija numpy-evega nakljucnega generatorja

Avtokorelacija RANDU nakljucnega signala



Slika 9: Avtokorelacijska funkcija RANDU nakljucnega generatorja