Matematično-fizikalni praktikum 2025/26

4. naloga: Fourierova analiza

Samo Krejan, 28231092

Pri numeričnem izračunavanju Fourierove transformacije

$$H(f) = \int_{-\infty}^{\infty} h(t) \exp(2\pi \mathrm{i} f t)\, \mathrm{d}t \tag{1}$$

$$h(t) = \int_{-\infty}^{\infty} H(f) \exp(-2\pi \mathrm{i} f t)\, \mathrm{d}f \tag{2}$$

je funkcija $h(t)$ običajno predstavljena s tablico diskretnih vrednosti

$$h_k = h(t_k), \quad t_k = k\Delta, \quad k = 0, 1, 2, \ldots N-1. \tag{3}$$

Pravimo, da smo funkcijo vzorčili z vzorčno gostoto (frekvenco) $f = 1/\Delta$. Za tako definiran vzorec obstaja naravna meja frekvenčnega spektra, ki se imenuje *Nyquistova frekvenca*, $f_c = 1/(2\Delta)$: harmonični val s to frekvenco ima v vzorčni gostoti ravno dva vzorca v periodi. če ima funkcija $h(t)$ frekvenčni spekter omejen na interval $[-f_c, f_c]$, potem ji z vzorčenjem nismo odvzeli nič informacije, kadar pa se spekter razteza izven intervala, pride do *potujitve* (*aliasing*), ko se zunanji del spektra preslika v interval.

Frekvenčni spekter vzorčene funkcije (3) računamo samo v N točkah, če hočemo, da se ohrani količina informacije. Vpeljemo vsoto

$$H_n = \sum_{k=0}^{N-1} h_k \exp(2\pi \mathrm{i} k n/N), \qquad n = -\tfrac{N}{2}, \ldots, \tfrac{N}{2}, \tag{4}$$

ki jo imenujemo diskretna Fourierova transformacija in je povezana s funkcijo v (1) takole:

$$H(\tfrac{n}{N\Delta}) \approx \Delta \cdot H_n.$$

Zaradi potujitve, po kateri je $H_{-n} = H_{N-n}$, lahko pustimo indeks n v enačbi (4) teči tudi od 0 do N. Spodnja polovica tako definiranega spektra ($1 \le n \le \frac{N}{2} - 1$) ustreza pozitivnim frekvencam $0 < f < f_c$, gornja polovica ($\frac{N}{2} + 1 \le N - 1$) pa negativnim, $-f_c < f < 0$. Posebna vrednost pri $n = 0$ ustreza frekvenci nič ("istosmerna komponenta"), vrednost pri $n = N/2$ pa ustreza tako f_c kot $-f_c$.

Količine h in H so v splošnem kompleksne, simetrija v enih povzroči tudi simetrijo v drugih. Posebej zanimivi so trije primeri:

če je	h_k realna	tedaj je	$H_{N-n} = H_n^*$
	h_k realna in soda		H_n realna in soda
	h_k realna in liha		H_n imaginarna in liha

(ostalih ni težko izpeljati). V tesni zvezi s frekvenčnim spektrom je tudi moč. *Celotna moč* nekega signala je neodvisna od reprezentacije, Parsevalova enačba pove

$$\sum_{k=0}^{N-1} |h_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |H_n|^2$$

(lahko preveriš). Pogosto pa nas bolj zanima, koliko moči je vsebovane v frekvenčni komponenti med f in $f + \mathrm{d}f$, zato definiramo enostransko spektralno gostoto moči (*one-sided power spectral density*, PSD)

$$P_n = |H_n|^2 + |H_{N-n}|^2 \,.$$

Pozor: s takšno definicijo v isti koš mečemo negativne in pozitivne frekvence, vendar sta pri realnih signalih h_k prispevka enaka, tako da je $P_n = 2\,|H_n|^2$.

Z obratno transformacijo lahko tudi rekonstruiramo h_k iz H_n

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n \exp(-2\pi \mathrm{i} k n/N) \tag{5}$$

(razlika glede na enačbo (4) je le predznak v argumentu eksponenta in utež $1/N$).

1 Naloga

1. Izračunaj Fourierov obrat Gaussove porazdelitve in nekaj enostavnih vzorcev, npr. mešanic izbranih frekvenc. Za slednje primerjaj rezultate, ko je vzorec v intervalu periodičen (izbrane frekvence so mnogokratniki osnovne frekvence), z rezultati, ko vzorec ni periodičen (kako naredimo Gaussovo porazdelitev 'periodično' za FT?). Opazuj pojav potujitve na vzorcu, ki vsebuje frekvence nad Nyquistovo frekvenco. Napravi še obratno transformacijo (5) in preveri natančnost metode. Poglej, kaj se dogaja z časom računanja - kako je odvisen od števila vzorčenj?

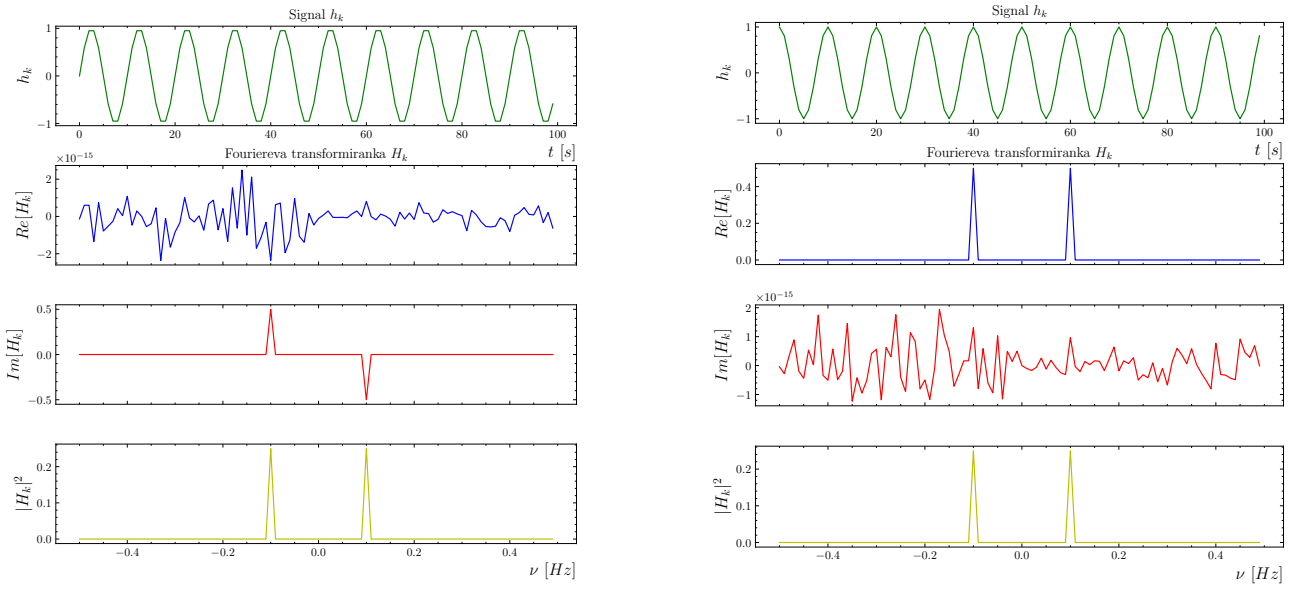
2. Po Fourieru analiziraj 2.3 s dolge zapise začetka Bachove partite za violino solo, ki jih najdeš na spletni strani Matematičnofizikalnega praktikuma. Signal iz začetnih taktov partite je bil vzorčen pri 44 100 Hz, 11 025 Hz, 5512 Hz, 2756 Hz, 1378 Hz in 882 Hz. S poslušanjem zapisov v formatu `.mp3` ugotovi, kaj se dogaja, ko se znižuje frekvenca vzorčenja, nato pa s Fourierovo analizo zapisov v formatu `.txt` to tudi prikaži.

1.1 Osnovni primeri

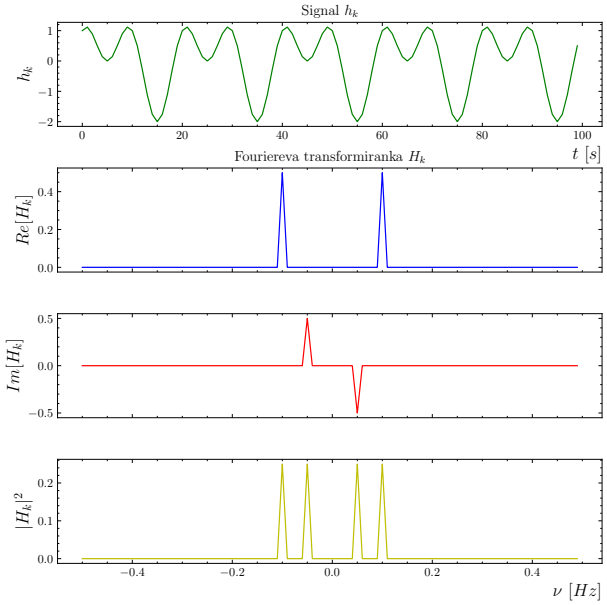
Zaradi časovne zahtevnosti Fourierove transformacije sem se odločil za implementacijo DFT algoritma v RUST programskem jeziku. Metodo sem nato najprej preveril na nekaterih osnovnih primerih, kjer iz teorije vem kaj pričakovati. To sta bila v prvi vrsti čisti realni sinusni in kosinusni signal, kjer sem med drugim namenil veliko pozornosti temu, da sem poskrbel, da je signal vzorčen periodično: torej, če bi kopijo signala "prilepil"na njegovo levo ali desno stran ne bi bilo opaziti nobenih skokov. Za to sem poskrbel tako da sem *numpy* funkciji linspace dodal parameter $endpoint = False$. Na ta način sem dobil grafe (1, 2)

Na slik 1 vidimo že vse lastnosti Fourierjevega obrata, ki so opisane v uvodu. Ker je kosinusni signal realen in sod je tak tudi Fourierev obrat in ker je sinusni signal realen in lih, je njegov Fourierev obrat imaginaren in lih.

Slika 2 je super ponazoritev uporabnosti Fourierjevega obrata, saj smo z njim uspeli nek *čuden* signal dekonstruirati nazaj v njegove harmonične komponente

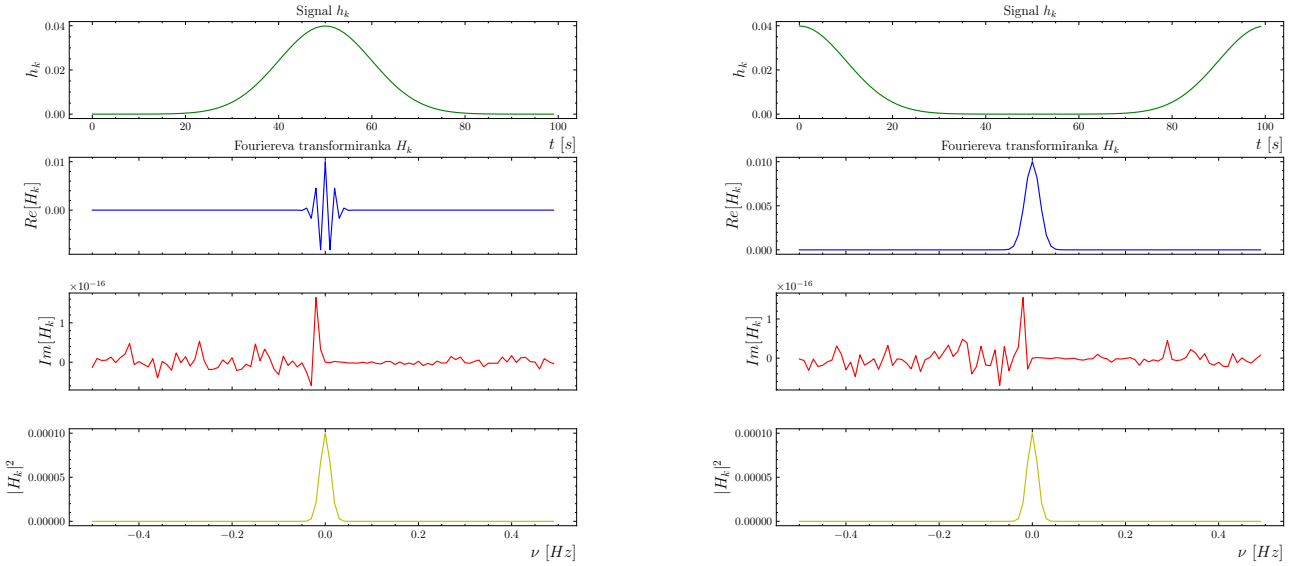


Slika 1: Fourierev obrat H_k sinusnega signala h_k (levo) in cosinusnega signala h_k (desno)



Slika 2: Fourierev obrat H_k kombinacije sinusnega in kosinusnega signala h_k z različnima frekvencama

Naslednji pomemben osnoven primer, ki sem si ga pogledal je DFT Gaussove funkcije. Ta je nekoliko specifičen, saj če ga želimo narediti periodičnega v smislu da je prva točka vzorčenja tista, ki ustreza vrednosti $x = 0$, moramo vzorčenje pravzaprav presekati na polovici in nato desno polovico premakniti na levo stran, levo pa na desno. Temu je tako, saj DFT predvideva, da se naše vzorčenje vendo začne v izhodišču. Razlika med upoštevanjem tega detajla in ne je prikazana na sliki 3

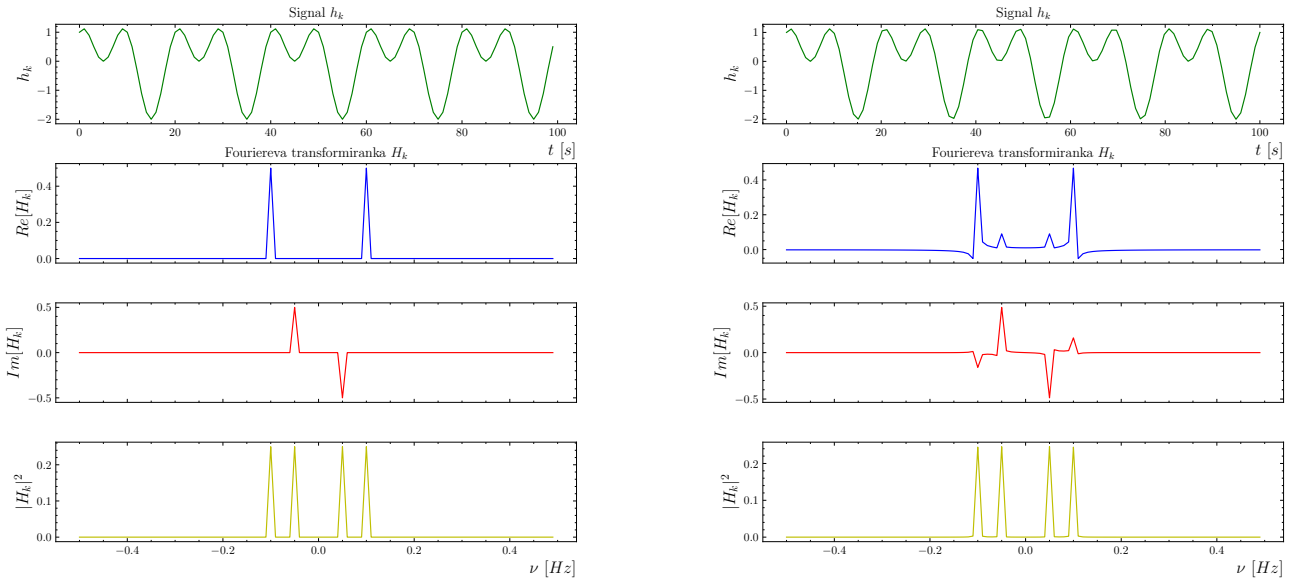


Slika 3: Fourierev obrat H_k Gaussove funkcije h_k ki je periodična (desno) in ne-periodična (levo)

Vidimo, da na levem delu slike 3 pri Fourierevem obratu pride do nekih "dodatnih sinusov"kar se da teoretično razložiti kot Fourierjev obrat Gaussove funkcije prestavljene v desno, na desni pa vidimo (kot je pričakovati), da je obrat Gaussove funkcije Gaussova funkcija.

1.2 Problem neperiodičnosti

V poglavju 1.1 sem omenil, da sem namenil kar nekaj pozornosti namenil ohranjanju periodičnosti signala (tako, da sem uporabil zastavico *endpoint* = *False* pri numpy linspace). A kaj bi se zgodilo, če na to ne bi bil pozoren? Izkaže se, da pride do pojava *puščanja* (ang: leakage). Zaradi le tega višine vrhov niso več sorazmerne z jakostjo signala in sam signal ni več omejen na eno samo frekvenco. Vizualno se to lepo vidi na skili 4, kjer je signalu iz slike 2 dodana zgolj ena točka, ki uniči periodičnost a zaradi nje že vidimo puščanje.



Slika 4: Fourierev obrat H_k kombiniranega signala (glej sliko 2) h_k ki je periodična (desno) in ne-periodična (levo)

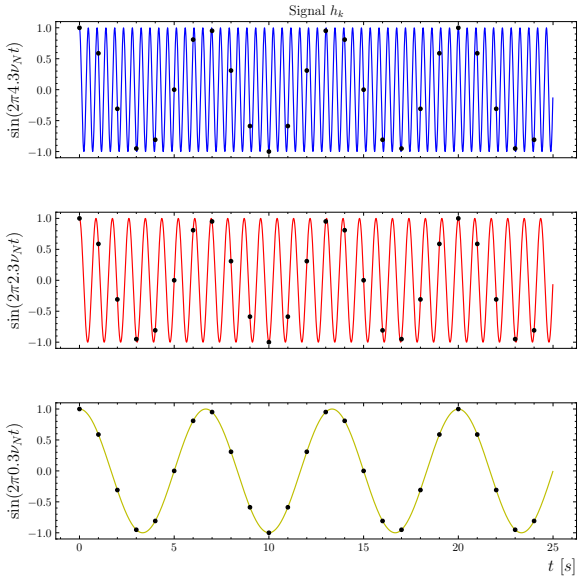
Vidimo, da na srečo težava s periodičnostjo ne vpliva na $|H_k|^2$, tako kot smo to lahko opazili že pri

Gaussovi funkciji na sliki 3

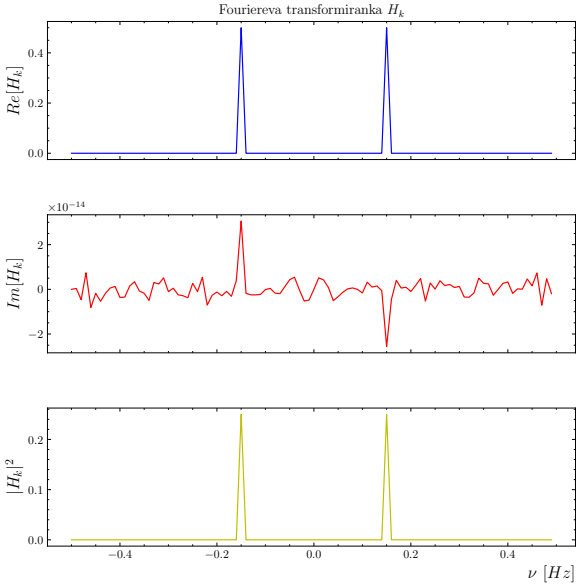
1.3 Potujitev

Do naslednjega izziva sem prišel, ko sem poskusil narediti Fourierev obrat za harmonične signale s frekvenco višjo od Nyquistove frekvence (ν_N). Ko sem recimo skušal narediti obrat signala s frekvenco enako $2,3\nu_N$ sem dobil transformacijo ekvivalentno transformaciji signala s frekvenco $0,3\nu_N$.

Izkaže se, da je temu tako, saj če signal vzorčimo s frekvenco $2\nu_N$ je vzor čenje ekvivalentno za signale s frekvencami $\nu + 2k\nu_N;\ k \in \mathbb{N}$. To je lepo ponazoreno na sliki 5



Slika 5: Grafi kosinusnih signalov s frekvencami (od spodaj navzgor) $0,3\nu_N$, $0,3\nu_N+2\nu_N$, $0,3\nu_N+4\nu_N$. Vidimo, da če funkcijo vzorčimo s frekvenco $2\nu_N$ dobimo pri vseh signalih enake vrednosti (črne točke)



Slika 6: Fourierev obrat ene izmed funkcij na sliki 5. Zaradi potujitve je nemogoče vedeti katero izmed funkcij smo vzorčili

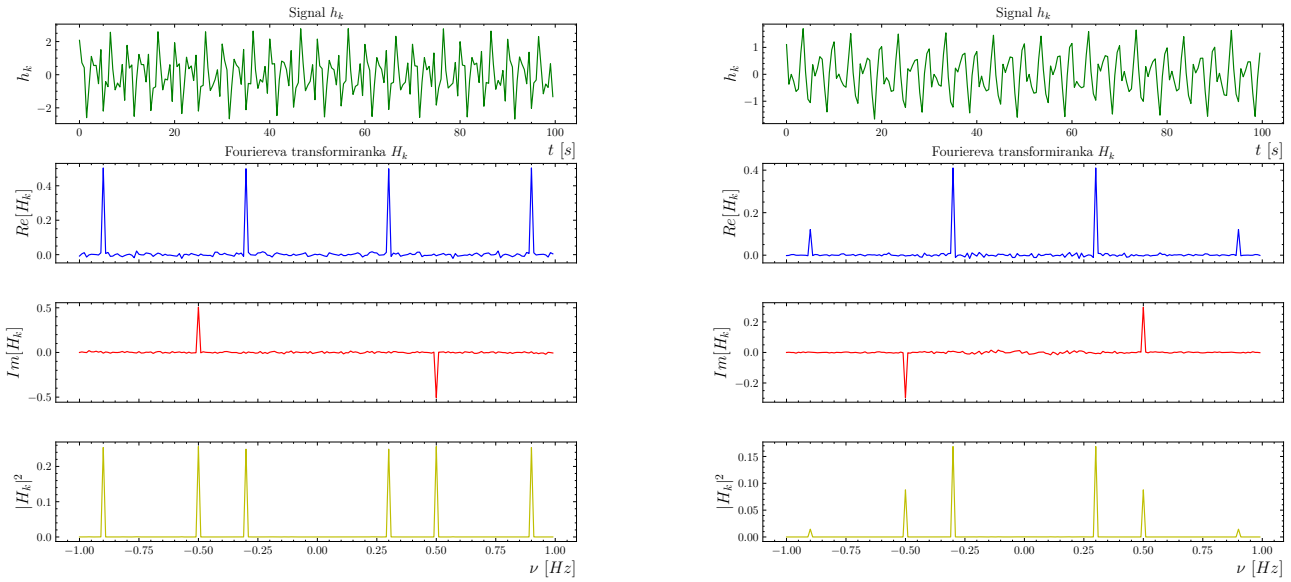
1.4 Filtriranje

Kot smo videli, je potujitev zahrbten pojav, zaradi katerega nam višje frekvence (velikokrat se pojavijo kot šum) uničijo spekter signala ki ga iščemo (več o tem bomo videli kasneje). Zaradi tega razloga se pri eksperimentih kjer nas zanima spekter odziva ponavadi uporablja vzorčenje pri zelo visokih frekvencah. A večinokrat nas visoke frekvence ne zanimajo in se jih za to želimo znebiti.

To lahko storimo z visokofrekvenčnim filtrom (največkrat implementiranim "hardversko"v spektrometru), ki preden se izvede DFT, zgladi signal tako, da naredi konvolucijo signala z Gaussovo funkcijo z $\sigma = 1/f_f$, kjer je f_f frekvenca filtra, ki je tipično enaka nekaj ν_N. Učinkovitost filtriranja je prikazana na sliki 7

1.5 Natančnost in hitrost implementacije

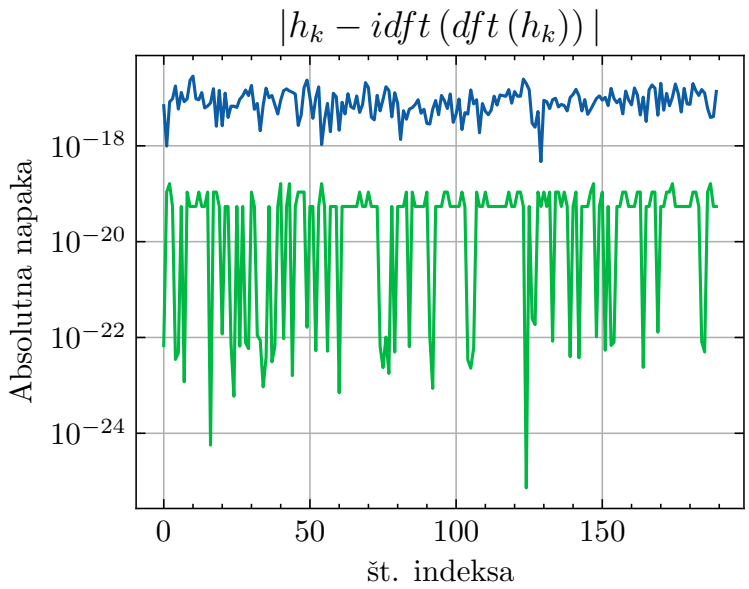
Do te točke sem samozavestno uporabljal lastno implementacijo, na tej točki pa se sprašujem, če je le to opravičeno. Uspešna implementacija mora biti natančna in sorazmerno hitra.



Slika 7: Fourierev obrat H_k kombiniranega in zašumljenega signala h_k brez filtra (levo) in s filtrom (desno). Vidimo, da so visoke frekvence močno "porezane", medtem ko so nizke skoraj nespremenjene.

1.5.1 Natančnost

Natančnost implementacije sem preveril tako, da sem vzel Gaussovo funkcijo, jo Fourierovo transformiral (DFT) in nato naredil še obratno transformacijo (iDFT). Napaka implementacije sem nato ocenil kot razliko med originalnim signalom in dvakrat transformiranim. Dobljeno natančnost sem primerjal z na enak način ocenjeno natančnostjo *numpy*-jeve implementacije "fft". Rezultate sem prikazal na grafu 8

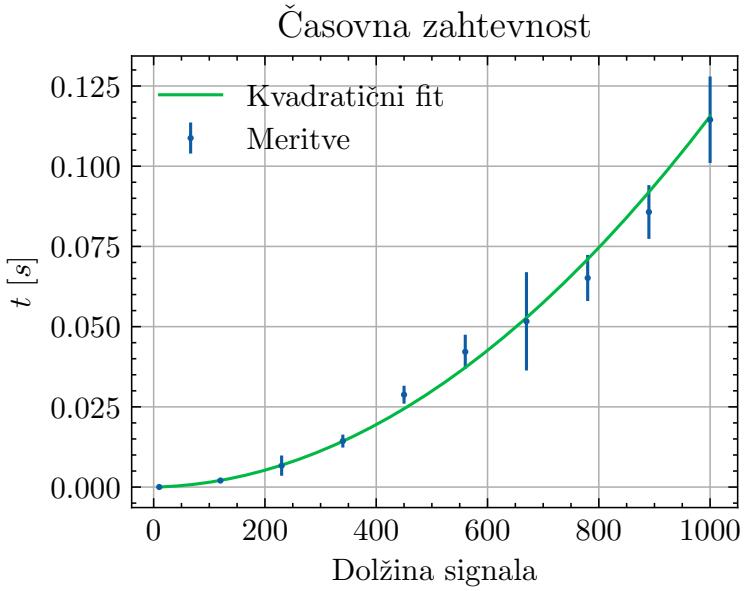


Slika 8: Primerjava natančnosti lastne implementacije in *numpy*-jeve

Čeprav je *numpy*-jeva implementacija skoraj dva velikostna reda bolj natančna, se tudi natančnost lastne implementacije giblje okoli numerične natančnosti računalnika, tako da osebno sem precej zadovoljen z dobljenim.

1.5.2 Hitrost

V teoriji je časovna zahtevnost DFT algoritma kvadratična, kar sem preveril tako, da sem za različne dolžine signala dvajsetkrat izvedel lastno implementacijo DFT in vsakič beležil potreben čas. Tako sem lahko ocenil povprečen čas za izvedbo DFT kot pa tudi napako le tega. Na "izmerjene"podatke sem na to fital kvadratno funkcijo $t = an^2 + bn + c$ in dobil vrednosti $a = (1.11 \pm 0.07) \cdot 10^{-7}\ s$, $b = (4 \pm 2) \cdot 10^{-6}\ s$ in $c = (-3 \pm 2) \cdot 10^{-5}\ s$. Uspešnost fita je skupaj s podatki prikazana na sliki 9



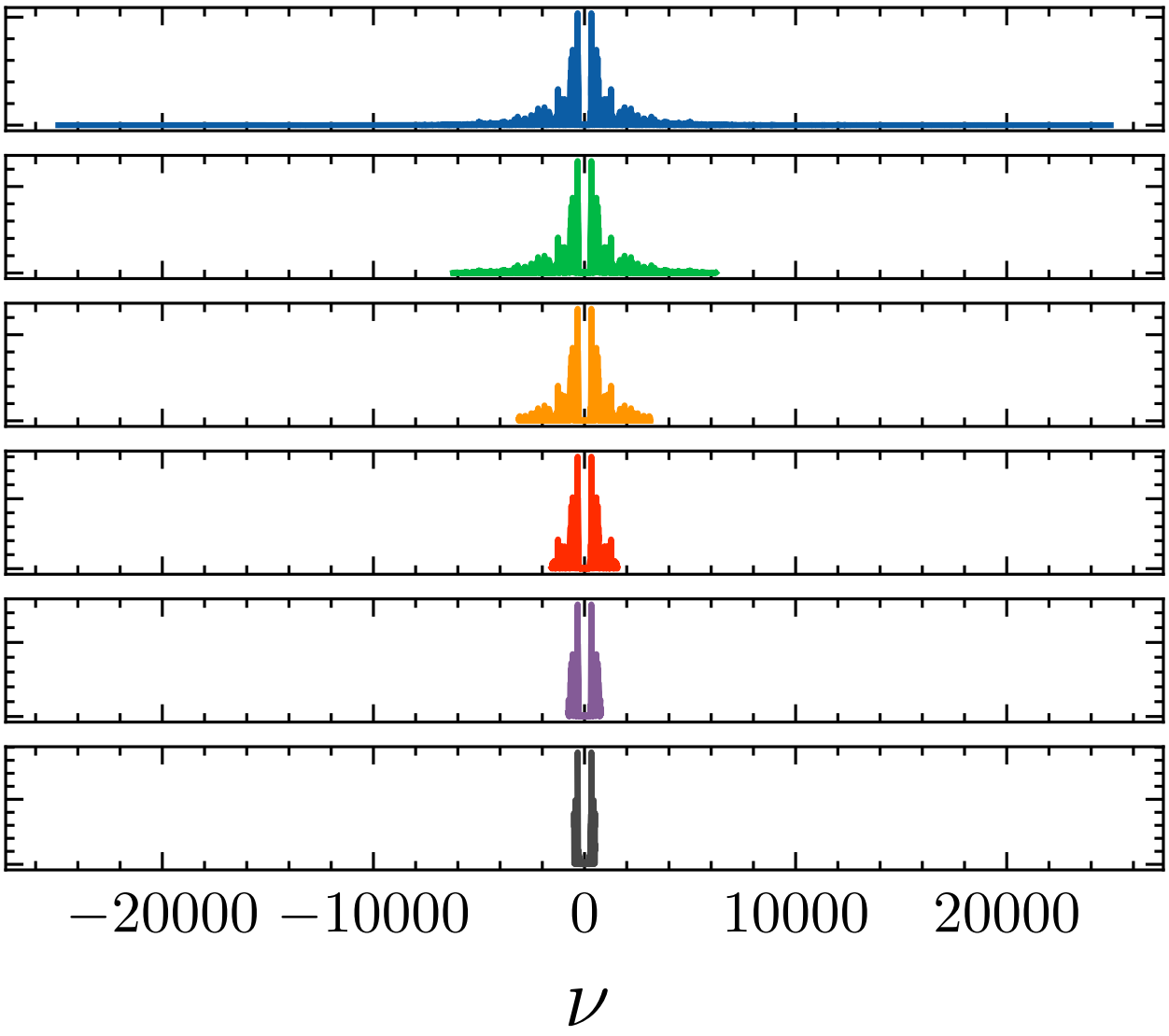
Slika 9: Časovna zahtevnost DFT

1.6 Bach

Del naloge je bil poslušati in kasneje analizirati zvočne posnetke Bachove partite za violinin solo, pri različnih frekvencah vzorčenja.

Že pri poslušanju se je opazilo (slišalo), da vzorčenje pri višji frekvenci ponuja bolj bogat zvok, kjer slišimo tudi višje frekvence, medtem ko vzorčenje z zelo nizkimi frekvencami zveni skoraj zašumljeno. Iz prej naučenega sem že med poslušanjem sklepal, da slišim posledice učinka potujitve. To seveda lahko preverim le tako, da izvedem Fourierev obrat vseh posnetkov in si ogledam v čem se spektri razlikujejo. Na ta način sem dobil graf 10

Na grafu res dobro vidimo kako so v spektru vidne višje frekvence le če vzorčimo z dovolj visoko frekvenco. Sicer sem izbrisal skalo na y osi zaradi preglednosti, a tam se je dalo videti tudi, da se same višine spektrov močno razlikujejo. Temu je tako saj je vrjetno prišlo do potujitve in so se višje frekvence (če je bilo vzorčenje prepočasno) preslikale nižje na spekter in tako spremenile vrednosti na tistih točkah. Potujitev pojasni tudi slišano zašumljenost/popačenost glasbe.



Slika 10: Spekter Bachove partite posnete z vzorčenjem s frekvencami (od zgoraj navzdol) 44 100 Hz, 11 025 Hz, 5512 Hz, 2756 Hz, 1378 Hz in 882 Hz

2 Dodatna naloga

Ker imam iz praktikumskih vaj travme sem se odločil da bom namesto, da analiziram resonator že drugič, raje analiziral glasbo, ki jo poslušam vsak dan. S tem razlogom sem se odločil da naredim preprost vizualizator glasbe (z mnogo pomoči Chat-GPTja), s katerim sem nato med drugim analiziral pesem Back in Black izvajalca AC/DC.

Posnetek demonstracije uporabe je (upam da) pripet zraven tega poročila. Če pa bi se kdo, ki bere to poročilo odločil vizualizator preizkusiti s še kakšno drugo glasbo (ki mora biti v 16bit 1 channel wav formatu) lahko kodo in navodila za uporabu najde na mojem githubu

3 Zaključek

Na tej točki moram priznati, da imam v resnici že nekaj predhodnih izkušenj z Fourierovo transformacijo, saj sem na IJS napisal program za zbiranje in obdelavo podatkov NMR eksperimentov.

Kljub temu, se mi zdi, da sem se tekom naloge naučil marsikaj novega. Predvsem se še nikoli doslej nisem lotil samostojne implementacije DFT in pa uporabe le te na sintetično pridobljenih signalih oziroma na glasbi. Sploh slednje je v meni vzbudilo kar nekaj zanimanja, tako da sem se posledično naučil veliko tudi o zgodovini $44,1kHz$.

Poleg tega sem nekoliko izostril svoje *matplotlib* sposobnosti (upam da se opazi), čeprav me je kasneje, ko sem slike želel vstaviti v LaTeX začela boleti glava, saj slike niso šle kamor sem si želel (verjetno se opazi).

Mislim, da lahko nalogo razglasim kot uspešno.