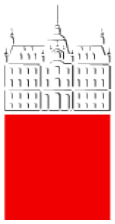
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Fourierova analiza

4. naloga pri Matematično-fizikalnem praktikumu

Avtor: Marko Urbanč (28191096)
Predavatelj: prof. dr. Borut Paul Kerševan

3.11.2021

Kazalo

1 Uvod

Fourierova transformacija je matematična transformacija, ki časovno ali prostorsko odvisno funkcijo "pretvori"v funkcijo, ki je odvisna od neke frekvence. Dober primer je Fourierova transformacija akorda sestavljenega iz več tonov, ki nam vrne frekvence in amplitude oz. glasnosti sestavnih tonov. Fourierovo transformacijo definiramo preko kompleksnega integrala:

$$\hat{f}(\nu) = \int_{-\infty}^{\infty} f(t) \exp(2\pi \mathrm{i}\nu t)\, \mathrm{d}t\ .$$

Funkcijo lahko iz frekvenčnega prostora oz. Fourierove slike pretvorimo nazaj v časovno domeno preko *inverzne Fourierove transformacije*, ki je podana z integralom:

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

Funkcijo $f(t)$ običajno predstavimo s tabelico diskretnih vrednosti:

$$f_k = f(t_k), \quad t_k = k\Delta, \quad k = 0, 1, 2, \ldots, N-1\ ,$$

kjer je N število vzorcev in Δ časovni razmak med dvema sosednjima vzorcema. Tako smo našo funkcijo vzorčili z vzorčno frekvenco $\nu = 1/\Delta$. Obstaja naravna meja frekvenčnega spektra, ki ga lahko dobimo. To je *Nyquistova frekvenca*, $\nu_c = 1/(2\Delta)$. Če ima funkcija frekvenčni spekter omejen na interval $[-\nu_c, \nu_c]$, potem z vzorčenjem nismo izgubili nobene informacije. Kadar pa imamo spekter tudi izen tega spektra pride do *potujitve* (ang. *aliasing*), kjer se zunanji del spektra preslika nazaj v ta interval.
Pri numeričnemu izračunu teh integralov se moramo seveda omejiti na neke končne vsote. *Diskretno Fourierovo transformacijo* v numeriki vpeljemo kot vsoto:

$$\hat{f}_n = \sum_{k=0}^{N-1} f_k \exp(-2\pi \mathrm{i}kn/N), \qquad n = -\tfrac{N}{2}, \ldots, \tfrac{N}{2},$$

in kjer velja zveza:

$$\hat{f}(\tfrac{n}{N\Delta}) \approx \Delta \cdot \hat{f}_n.$$

Zaradi potujitve je $hat f_{-n} = \hat{f}_{N-n}$ lahko pustimo indeks n v prejšnji vsoti, da teče tudi od 0 do N.
Inverzno Fourierovo transformacijo vpeljemo podobno kot:

$$f_k = \frac{1}{N} \sum_{n=0}^{N-1} \hat{f}_n \exp(2\pi \mathrm{i}kn/N)\ .$$

Tu moramo posebaj paziti na utež $1/N$, da dobimo pravilno transformacijo. Količine f in $\hat{f}$ 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

Celotna moč nekega signala je po Parsevalovi enačbi (1) neodvisna od reprezentacije.

$$\sum_{k=0}^{N-1} |f_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |\hat{f}_n|^2 \tag{1}$$

Za realne signale sta prispevka f_k enaka za $\pm\nu$. Zato, če nas zanima koliko moči je vsebovane v frekvenčni komponenti med ν in $\nu + \mathrm{d}\nu$, lahko definiramo enostransko spektralno gostoto moči kot:

$$P_n = |\hat{f}_n|^2 + |\hat{f}_{N-n}|^2 \,.$$

Lahko pa gledamo dvostransko, kjer se zavedamo, da imamo pravzaprav le prezrcaljeno sliko in nam negativne frekvence ne prinesejo nobene relevantne informacije.

2 Naloga

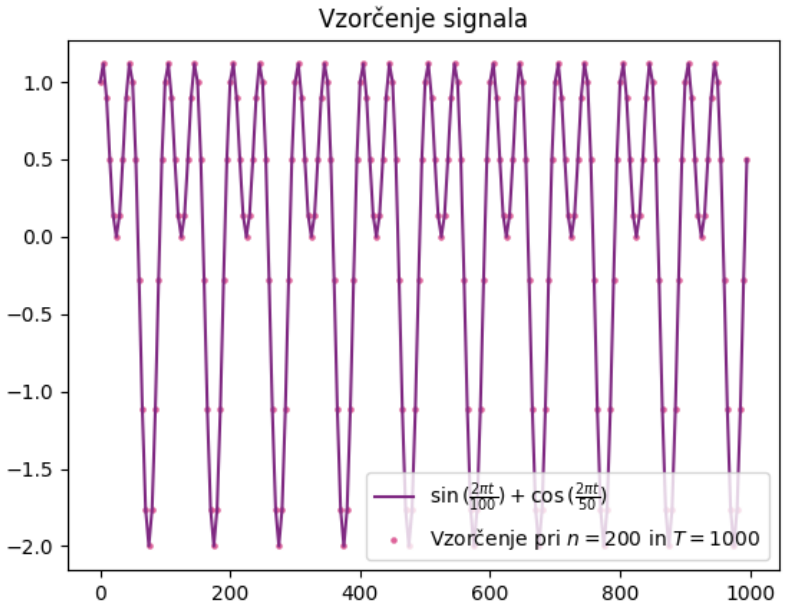
Naloga od nas zahteva, da izračunamo Fourierovo transformacijo Gaussove porazdelitve in nekaj preprostih vzorcev. Opazujemo in primerjamo transformacije, ko je vzorec periodičen in ko ni periodilen na intervalu. Želi, da opazujemo pojav potujitve na vzorcu, ki vsebuje frekvence nad Nyquistovo frekvenco. Preverimo še delovanje in natančnost obratne transformacije in časovno odvisnost računanja diskretne Fourierove transformacije v odvisnosti od števila vzorčenj. V končnem delu naloga zahteva, da analiziramo 2.3s dolge zapise Bachove partite za violino solo, ki so bile vzorčene pri različnih frekvencah vzorčenja, in ugotovimo, kaj se dogaja pri zniževanju le te.

3 Opis reševanja

Kot je navada, sem se problema lotil v Pythonu, kjer sem si spet veliko pomagal z knjižnicami `NumPy`, `SciPy` in `matplotlib`. Poskusil sem spisati osnovni funkciji za diskretno Fourierovo in inverzno diskretno Fourierovo transformacijo. Napisane funkcije so praktično identične kot `dft_slow(x)` iz zgleda na spletni učilnici. Napisal sem tudi funkciji, ki ustrezno za neka N in Δ pripravita časovni in frekvenčni interval.
Prvo sem se želel prepričati, da moj predpis za transformacijo deluje, zato sem preveril natančnost proti vgrajeni `np.fft.fft(x)` za nek preprost signal, definiran kot:

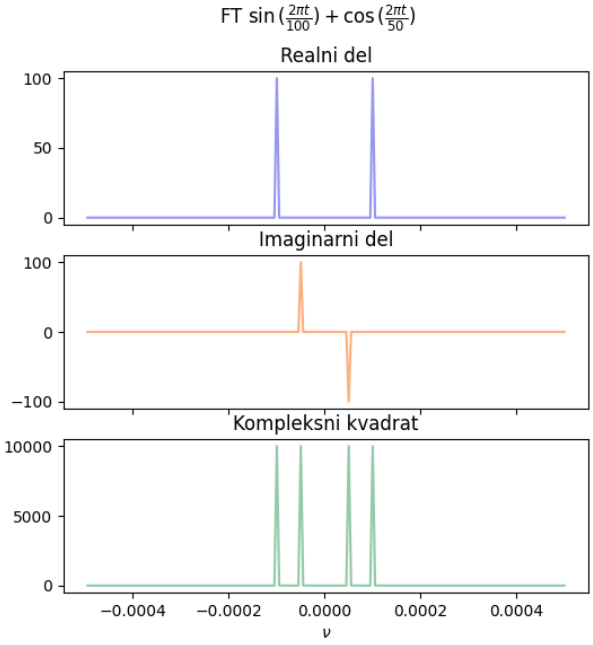
$$s(t) = \sin\left(\frac{2\pi t}{100}\right) + \cos\left(\frac{2\pi t}{50}\right).$$

Izkaže se, da je napisana funkcija proti NumPyjevi rutini, zelo primerljiva. Raz-

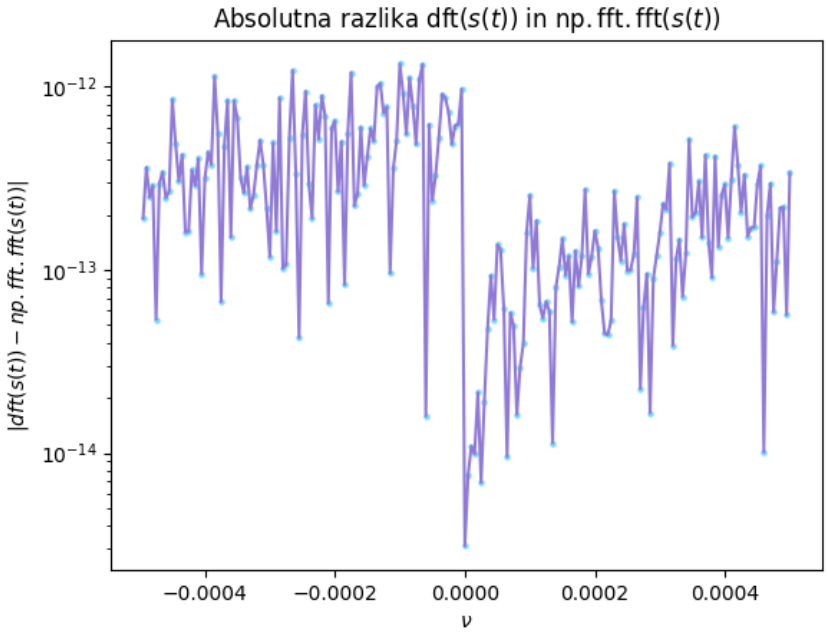


Slika 1: Signal $s(t)$ vzorčen pri $n = 200$ in $\Delta = T = 1000$.

like se poznajo šele pri 10^{-12}, kar je za naše potrebe zagotovo dovolj natančno.



Slika 2: Fourierova transformacija $\texttt{dft}(s(t))$ njene komponente in moč

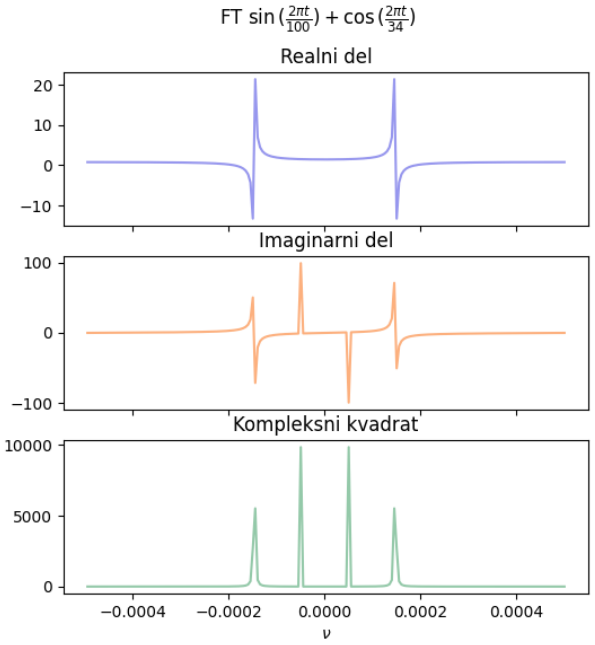


Slika 3: Primerjava z vgrajeno `Numpy` funkcijo

Zdaj imam zaupanje v svojo funkcijo in jo lahko mirno uporabljam (več o tem kasneje, ker ima vseeno težave z hitrostjo računanja). Pogledal sem si, kaj se zgodi, če vzorčim funkcijo, ki je sestavljena iz kotnih funkcij, kjer frekvence niso celoštevilski večkratniki. Vzorčen signal je:

$$s(t) = \sin\left(\frac{2\pi t}{100}\right) + \cos\left(\frac{2\pi t}{34}\right).$$

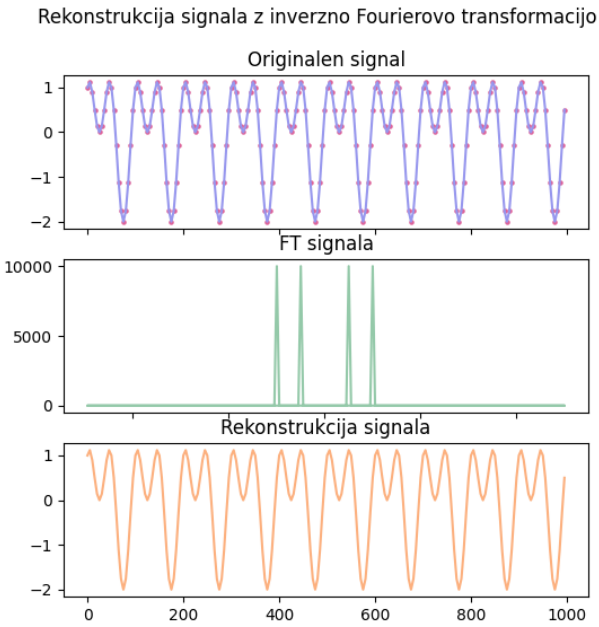
Opazimo, da je prišlo do razširitve črt in nekaj čudnosti v obeh komponentah



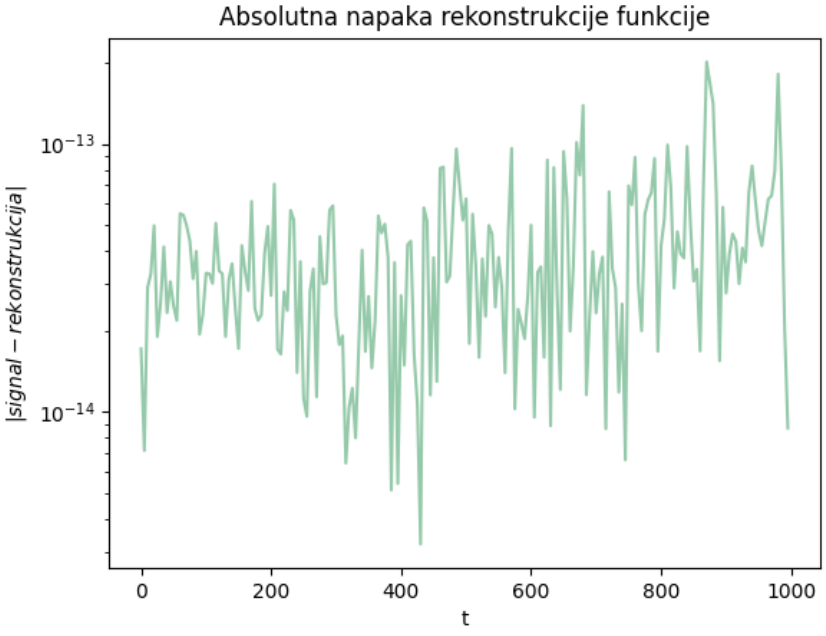
Slika 4: Fourierova transformacija `dft`$(s(t))$ vzorčenje $n = 200$ in $T = 1000$

transformacije.
Z inverzno Fourierovo transformacijo lahko tudi rekonstruiramo signal iz frekvenčnega spektra. Rekonstrukcijo sem poskusil za nekaj različnih signalov. Za vse sem tudi pogledal ujemanje z prvotnim signalom. Ujemanje je okoli meje napake, ki jo ima moja koda za transformacijo. Torej lahko iz spektra brez izgub rekonstruiramo signal. Signali:

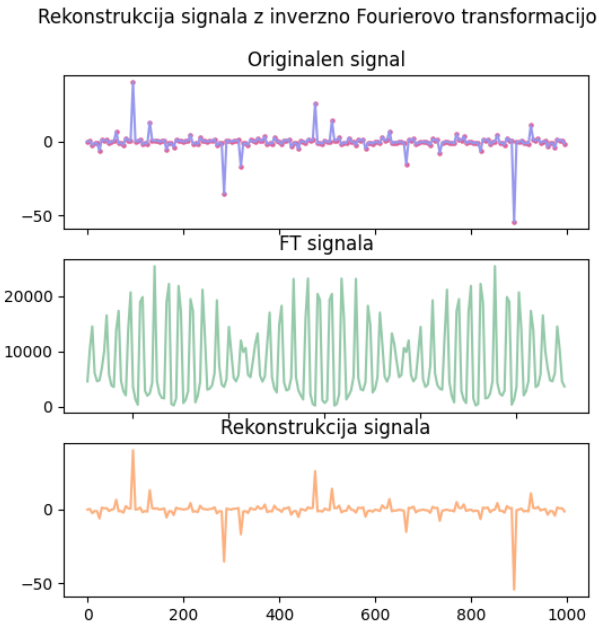
$$s_1(t) = \sin\left(\frac{2\pi t}{100}\right) + \cos\left(\frac{2\pi t}{50}\right),$$
$$s_2(t) = \sin\left(\frac{2\pi t}{100}\right) + \cos\left(\frac{2\pi t}{34.5607895}\right),$$
$$s_3(t) = e^{-t/50}\sin t\,.$$



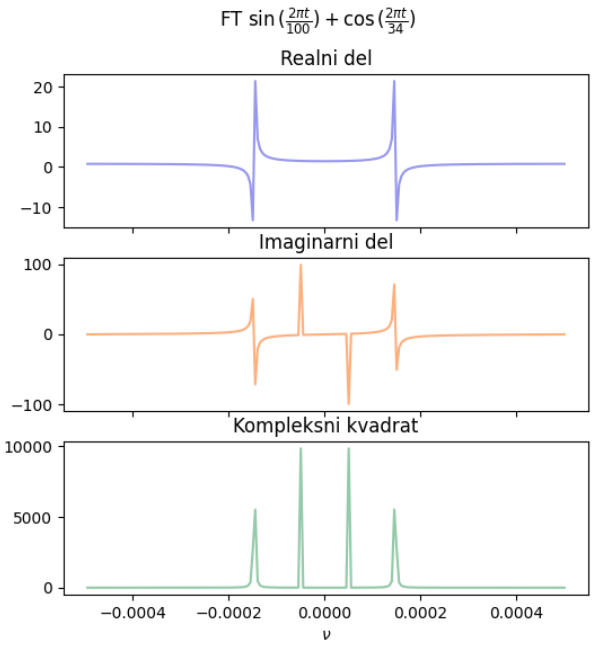
Slika 5: Rekonstrukcija z inverzno FT `dft`$(s_1(t))$ vzorčenje $n = 200$ in $T = 1000$



Slika 6: Absolutna razlika med originalnim signalom $s_1(t)$ in rekonstruiranim



Slika 7: Rekonstrukcija z inverzno FT $\texttt{dft}(s_2(t))$ vzorčenje $n = 200$ in $T = 1000$

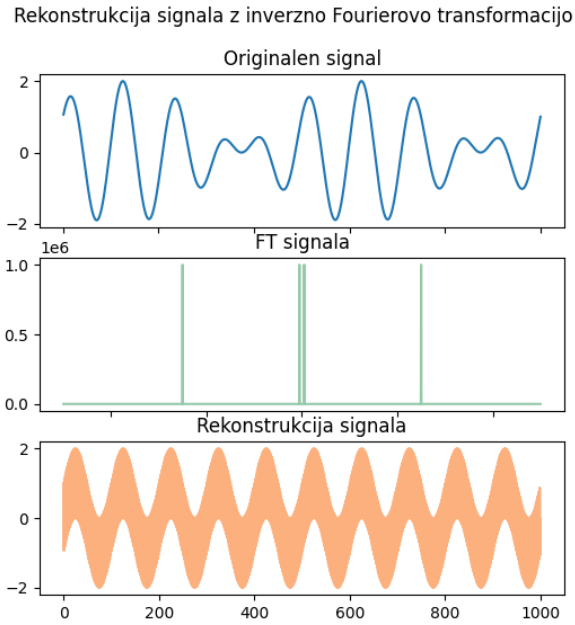


Slika 8: Rekonstrukcija z inverzno FT $\texttt{dft}(s_3(t))$ vzorčenje $n = 200$ in $T = 1000$

Po Nyquist-Shannonovem teoremu [1], ki pravi, da mora biti periodičen signal vzorčen z frekvenco vzorčenja, ki ima vsaj dvakrat večjo frekvenco kot največja komponente frekvence vzorčenega signala, rekonstrukcija signala, ki ima frekvenčne komponente nad Nyquistovo frekvenco ni več možna, ker pride do potujevanja. To sem preveril za signal:

$$s_4(t) = \sin\left(\frac{2\pi t}{100}\right) + \cos\left(\frac{2\pi t}{0.08}\right),$$

ki je bil vzorčen pri $n = 2000$ in $T = 1000$, kar nam da $\nu_c = 0.0005$. Rekonstrukcija signala ne uspe.

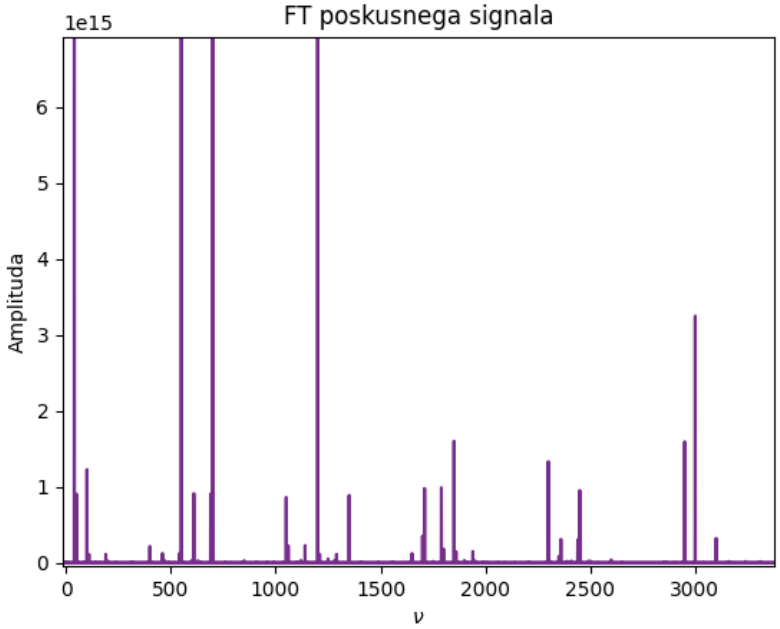


Slika 9: Rekonstrukcija `dft`$(s_4(t))$ s komponento nad ν_c vzorčenje $n = 2000$ in $T = 1000$

Zdelo se mi je tudi zanimivo napraviti animacijo, ko komponenta signala prečka ν_c, kar bom priložil ob temu poročilu. Nokoliko mi je sicer `matplotlib` povzročal težave, ker tako kot je koda za transformacijo zastavljena, najprej narišemo pozitivne frekvence in nato šele negativne, kar pomeni da dobimo kdaj čez graf kakšno čudno črto. V splošnem se mi je dalo temu izogniti, predvsem z upoabo raznih permutacij dobljenih arrayov. Pri animaciji pa kljub trudu nisem uspel odpraviti tega problema.

Za drugi del naloge pa sem z uporabo `scipy.io` knjižnice spisal funkcijo za Fourierovo analizo `.wav` zvokovnih datotek. Za format sem se odločil, ker je bil moj originalni namen obdelati nekaj podatkov opazovanj pulzarjev, ki so podani v `.wav`. Izkazalo se je, da so podatki bistveno preveč zašumljeni in bolj primerni za obdelavo pri naslednji nalogi.

Zdelo se mi je smiselno preveriti uporabo funkcije kakem poskusnem signalu, da se lahko prepričam v pravilno delovanje. V programu `Audacity` sem ustvaril mešanico sinusoidnih tonov pri različnih frekvencah in amplitudah. Vzorčil sem s frekvenco 44.1 kHz. Funkcija očitno dobro deluje. Lahko prepoznamo in razberemo osnovne frekvence tonov.

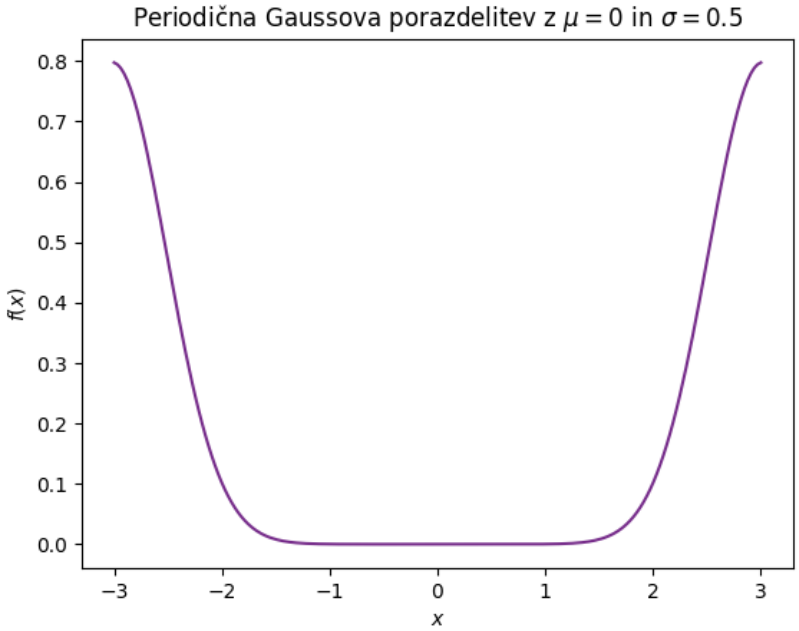


Slika 10: Testni signal iz sinusoidnih tonov 40, 550, 700, 1200 in 3000 Hz

4 Rezultati

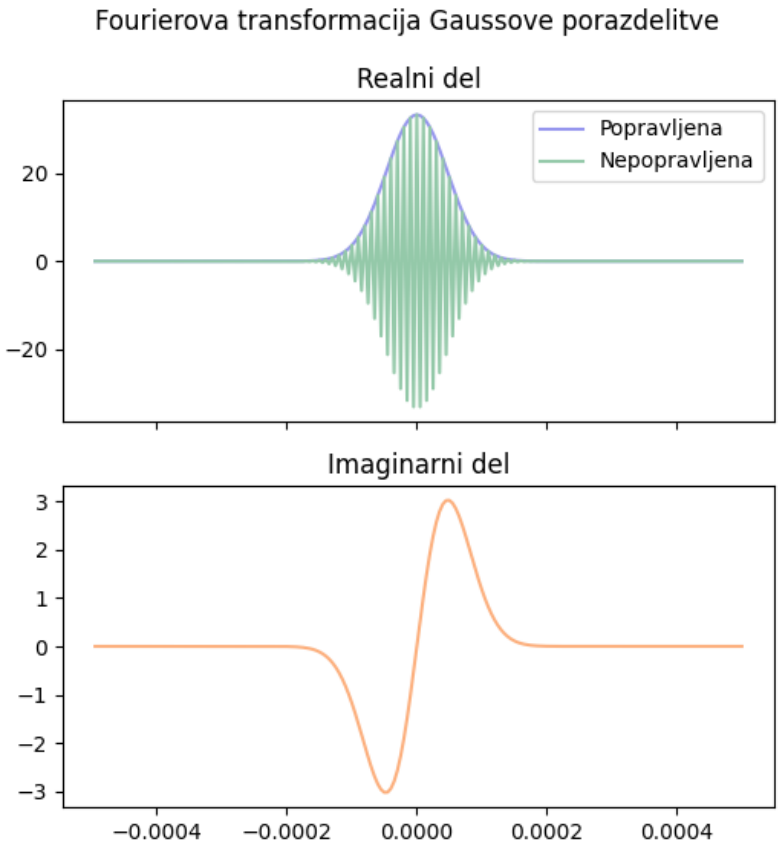
4.1 Fouriereva transformacija Gaussove porazdelitve

Naloga je od nas zahtevala, da naredimo Fourierovo transformacijo Gaussove porazdelitve. Zaplet pride, ker transformacija sklepa, da imamo periodičen signal, kar pa Gaussova porazdelitev ni. Z ustreznim zrcaljenjem lahko Gaussovo porazdelitev spravimo v "periodično"obliko.



Slika 11: Gaussova porazdelitev centrirana na izhodišče in z prezrcaljenim negativnim delom

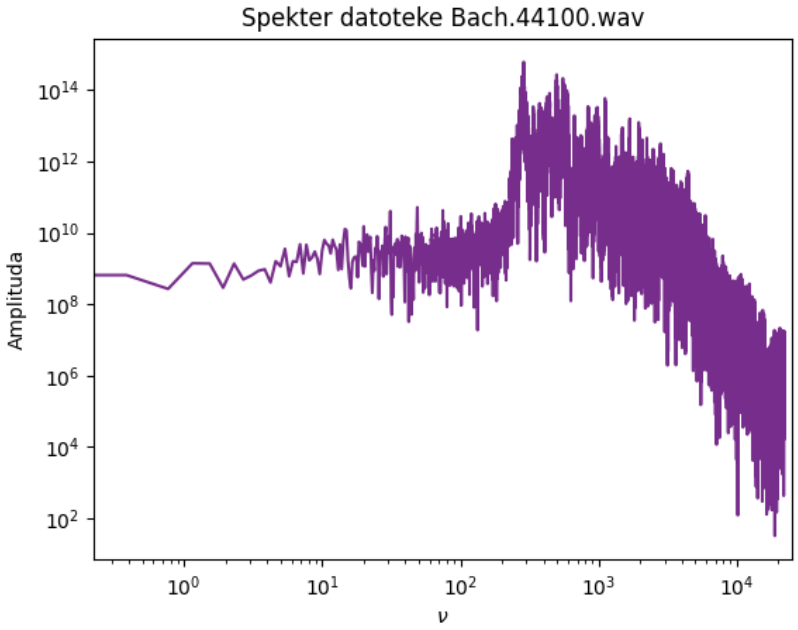
Fourierovo transformacijo pa lahko popravimo tudi eksplicitno tako, da transformiranko množimo z modulacijo $\exp \frac{2\pi i \nu T n}{2}$. Rezultat bo enak, kot če ustrezno prestavimo interval vzorčenja.



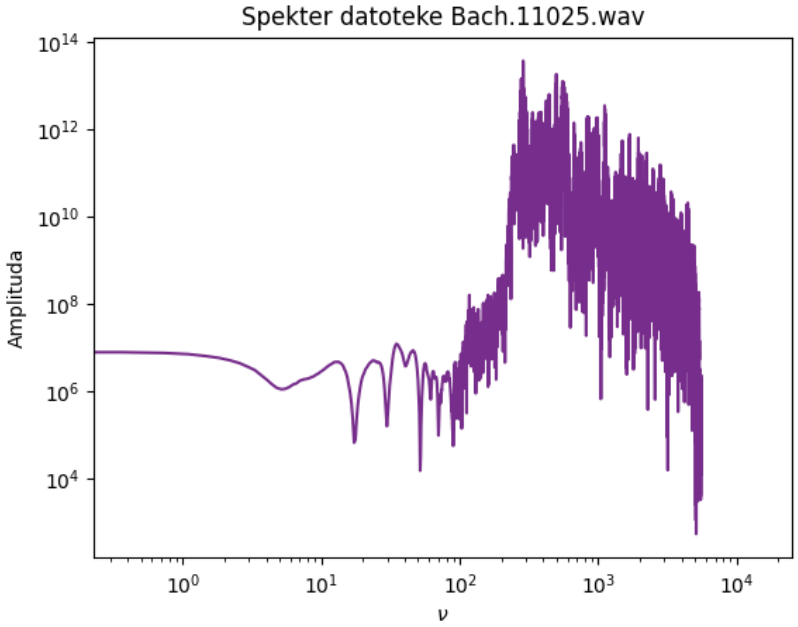
Slika 12: Fourierova transformacija Gaussove porazdelitve in njena popravljena oblika

4.2 Bachova partita za violino solo

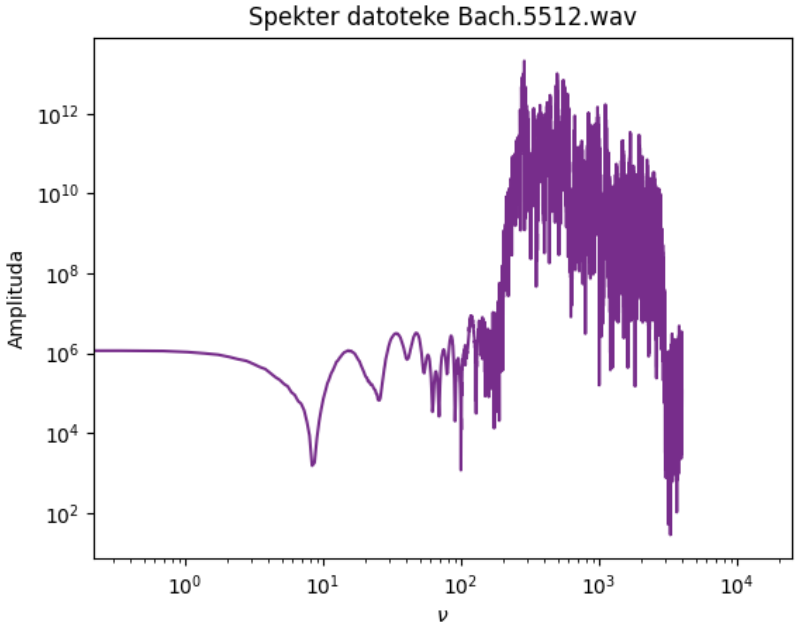
Dobili smo kratke posnetke pri različnih frekvencah vzorčenja. Te so 882 Hz, 1378 Hz, 2756 Hz, 5512 Hz, 11025 Hz in 44100 Hz. Po prej omenjenem Nyquistovem teoremu vemo, da nam frekvenca vzorčenja določa mejo najvišje frekvence, ki jo bomo lahko še reproducirali brez potujevanja. Slišimo lahko od okoli 20 Hz do 20 kHz. Zato je *CD kvaliteta* zvoka definirana kot vzorčenje pri 44.1 kHz. Tako imamo zagotovo vsaj dvakrat večjo frekvenco vzorčenja kot najvišje frekvence, ki jih lahko slišimo. V praksi zaradi končnega časa vzorčenja mora biti ta frekvenca celo malo več kot dvakrat najvišja frekvenca vzorčenega signala. Tu sem uporabil `np.fft.fft()` hitro Fourierovo transformacijo vsaj je lahko za 2 s dolg posnetek tudi čez 80000 vzorcev, kar pa navadno transformacijo dolgo vzame. Z hitrim poslušanjem posnetkov lahko to potrdimo. Nižja kot je frekvenca vzorčenja, manj jasen in čist je zvok. Zgubimo informacije o višjih frekvencah. Te se preslikajo znotraj intervala Nyquistove frekvence, kar nam povzroči zadušenost in slabo kvaliteto. To vse lahko tudi zelo dobro vidimo v frekvenčnem spektru. Zgleda kot da bi spekter filtrirali z low pass filtrom.



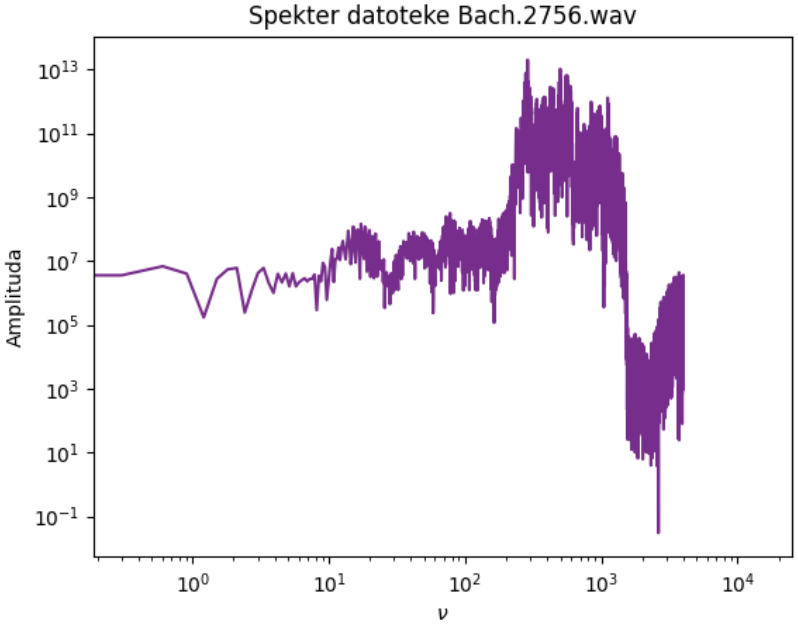
Slika 13: Spekter posnetka v logaritmični skali vzorčen pri $\nu = 44100$ Hz



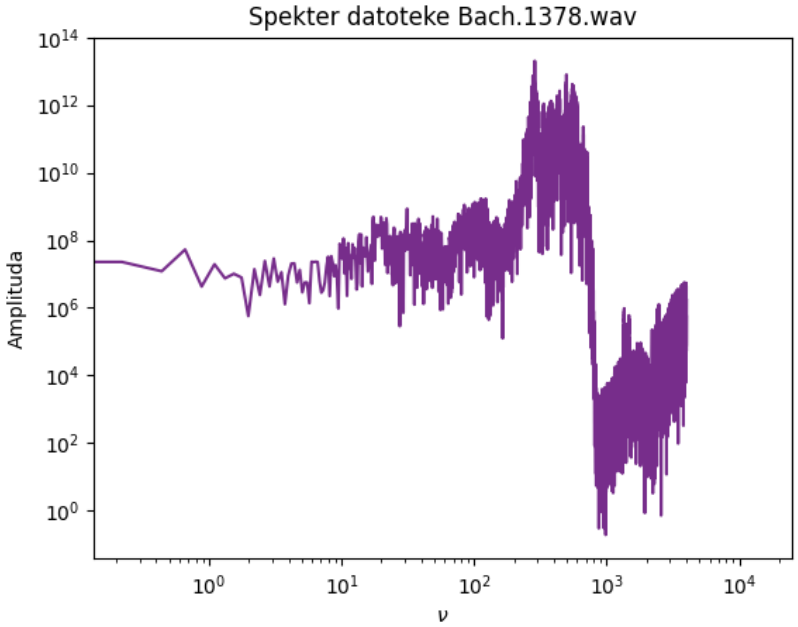
Slika 14: Spekter posnetka v logaritmični skali vzorčen pri $\nu = 11025$ Hz



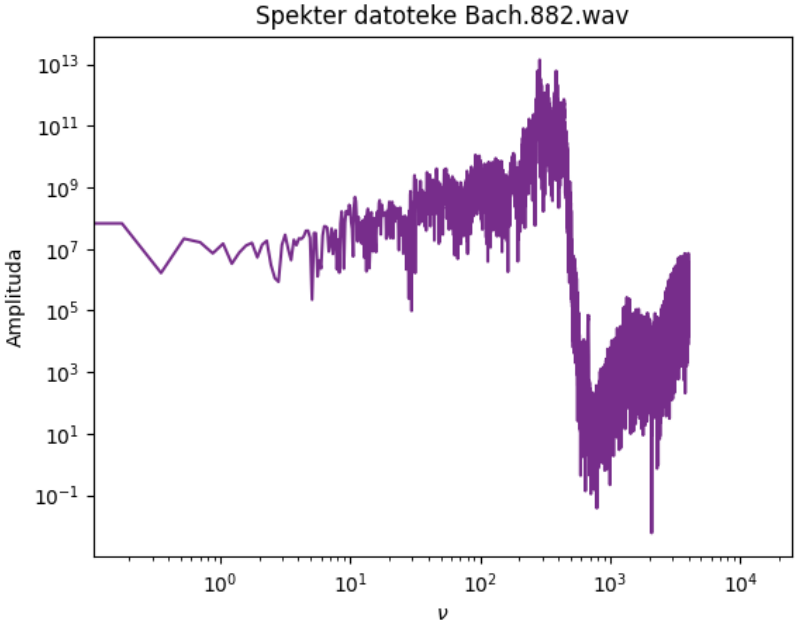
Slika 15: Spekter posnetka v logaritmični skali vzorčen pri $\nu = 5525$ Hz



Slika 16: Spekter posnetka v logaritmični skali vzorčen pri $\nu = 2756$ Hz



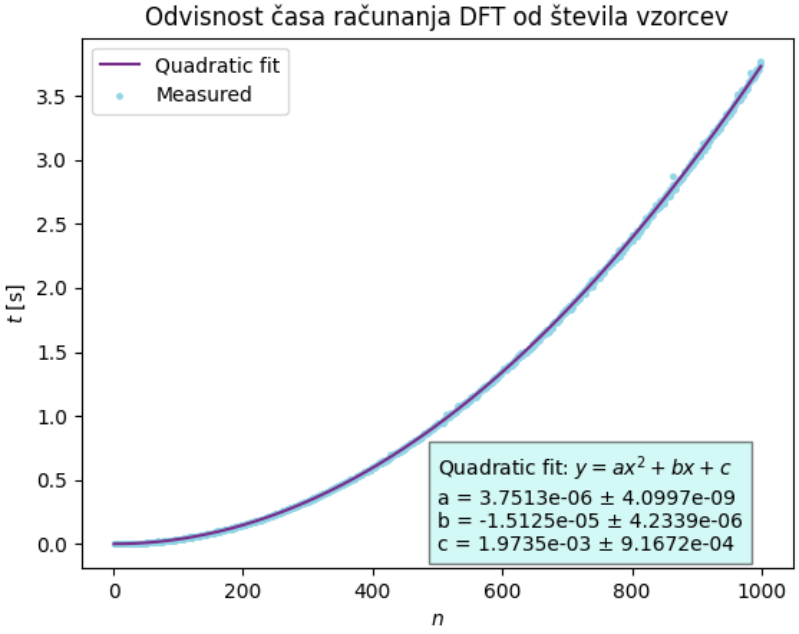
Slika 17: Spekter posnetka v logaritmični skali vzorčen pri $\nu = 1378$ Hz



Slika 18: Spekter posnetka v logaritmični skali vzorčen pri $\nu = 882$ Hz

4.3 Čas računanja DFT v odvisnosti od števila vzorcev

Ker sem sprva tako dolgo čakal na `dft(x)` pri obdelavi posnetkov (preden sem obupal in zamenjal na bliskovito hitri FFT), se mi je zdelo smiselno pogledati, kakšna je povezava med časom računanja in številom vzorcem. Izkaže se, da velja $t \propto n^2$. Do tega sem prišel preko merjenja časov in ustreznega prileganja različnih polinomov.

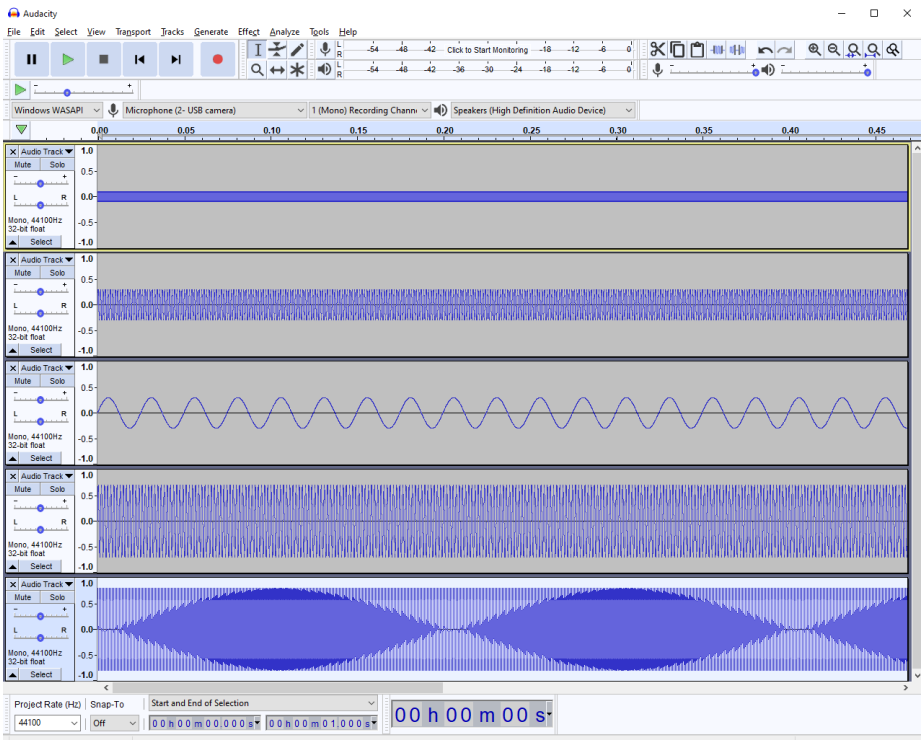


Slika 19: Kvadratična odvisnost časa računanja od števila vzorcev

5 Komentarji in izboljšave

Zdi se mi, da sem zopet večino pomembnih stvari komentiral že sproti. Dolgo časa sem se ukvarjal z transformacijami posnetkov pulzarjev za katere se izkaže, da so preveč zašumljeni, da bi lahko dobil kakšne smiselne rezultate. Mogoče si to lahko ogledam pri naslednji nalogi, kjer naj bi se tudi naučili nekoliko odpravljati šum. Vsekakor bi hitreje dobil nekatere grafe, če bi uporabljal vgrajeno FFT. Vseeno sem za vse razen posnetke ostal pri svoje verziji `dft(x)`, ker se mi je zdelo bolj primerno, da po svoje naredim. Zelo sem navdušen nad prilagojeno kvadratno funkcijo pri odvisnosti časa računanja. Vsekakor bi se pa lahko graf izboljšal z večjim območjem za n in z večimi ponovitvami in povprečenjem časov.

Želel sem si predstaviti še kaj delovanja sintesizerjev na princip frekvenčne modulacije, kjer se pojavijo čudovite stvari v spektru zvoka, ampak mi je žal popolnoma zmanjkalo časa, da bi snemal zvoke še za to obdelavo.
Imel sem tudi idejo, da bi v duhu vaje *Akustični resonator*, posnel frekvenčni odziv svoje tuš kabine. Med raznim *šamanskim* petjem sem ugotovil, da lahko zadanem nekaj resonančnih vrhov, kjer vse res očitno doni. Zopet škoda, da nimamo več časa, da bi lahko premislil postavitev tega eksperimenta. Vprašanje je tudi, kako bi bilo s šumom. Mogoče pa kdaj poskusim..



Slika 20: Generiranje testnih signalov v Audacity

Literatura

[1] Wikipedia. Nyquist–Shannon sampling theorem — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Nyquist%E2%80%93Shannon%20sampling%20theorem&oldid=1048443869, 2021. [Online; accessed 06-November-2021].