Spektralna analiza in filtriranje

Andrej Kolar-Požun, 28172042

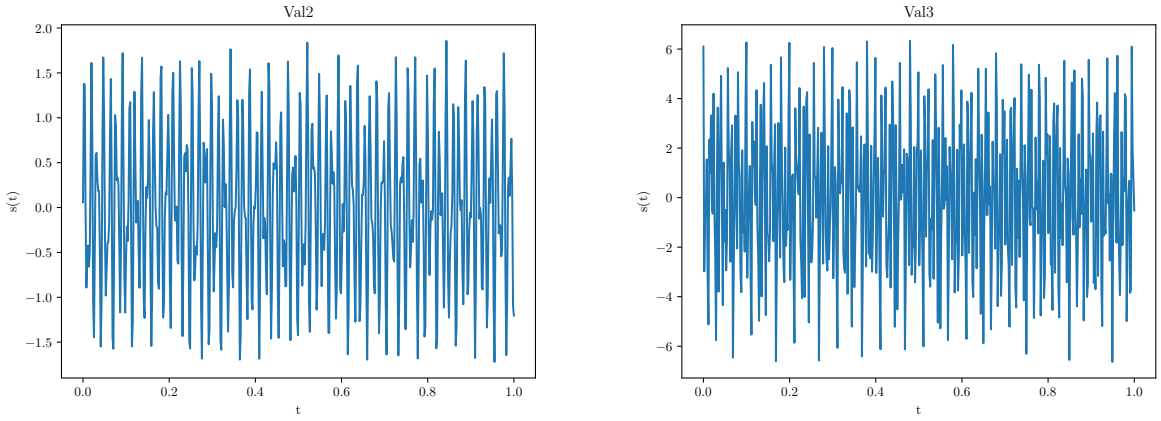
January 10, 2018

1 Spekter signalov

V tej nalogi bom spekter signalov določil s pomočjo diskretne Fourierove transformacije definirane na naslednji način:

$$A_k = \sum_{m=0}^{n-1} exp(-2\pi i \frac{mk}{n})$$

Moč frekvenčne komponente k je potem dana z $P_k = 0.5(|A_k|^2 + |A_{N-k}|^2)$. Tekom cele naloge bom uporabljal algoritem FFT iz Numpy-ja.



Signala, katera spekter bom analiziral. Diskretne točke sem zaradi boljše predstave povezal. Povprečna vrednost signala je reda velikosti stotinke, kar bom vseeno odštel, saj nas to ne zanima.Na tej sliki sem x skalo postavil od 0 do 1.

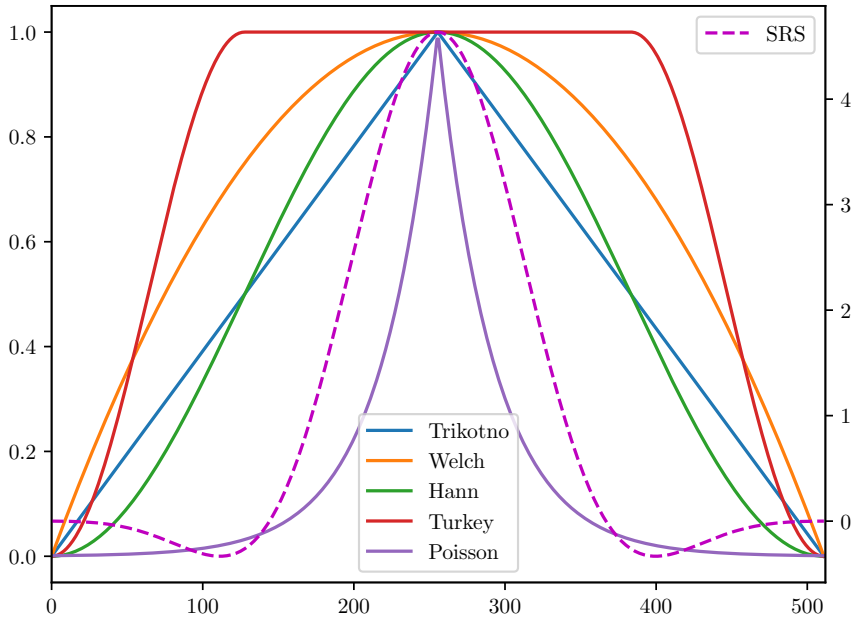
Na zgornjem prikazu se morda to ne vidi najbolje, a vrednost signala v začetni in končni točki ni enaka. Ker Fourierova transformacija predpostavlja periodičnost nam taki skoki lahko pokvarijo spekter. Zato se pred transformacijo ponavadi pomnoži z okenskimi funkcijami, ki signal na robu tako ali drugače postavijo na nič. Stestiral bom naslednje okenske funkcije:

$$Trikotno(n) = 1 - \left| \frac{n - 0.5(N-1)}{0.5(N-1)} \right|$$

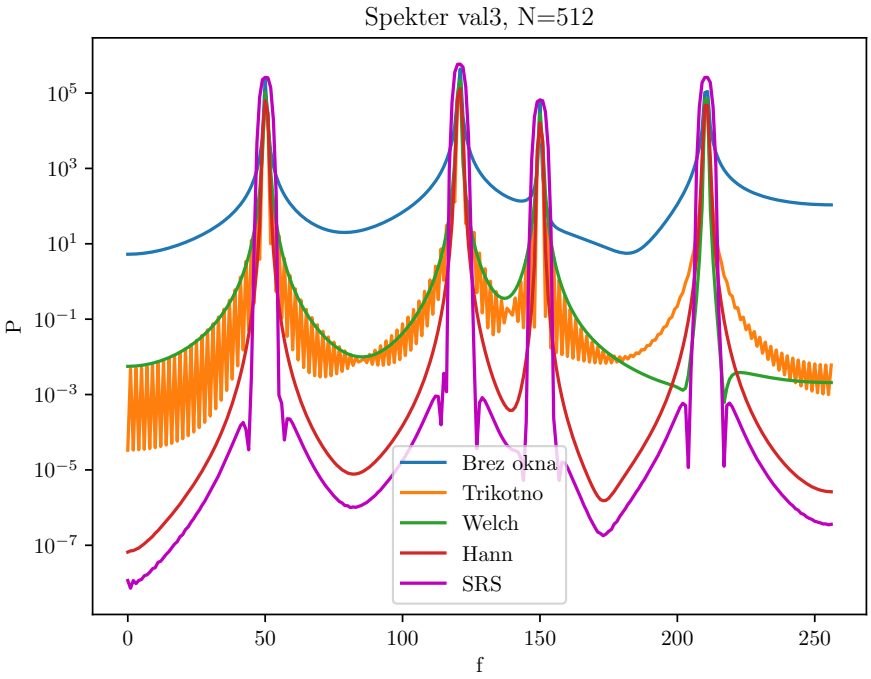
$$Welch(n) = 1 - \left(\frac{n - 0.5(N-1)}{0.5(N-1)} \right)^2$$

$$Hann(n) = sin^2 \left(\frac{\pi n}{N-1} \right)$$

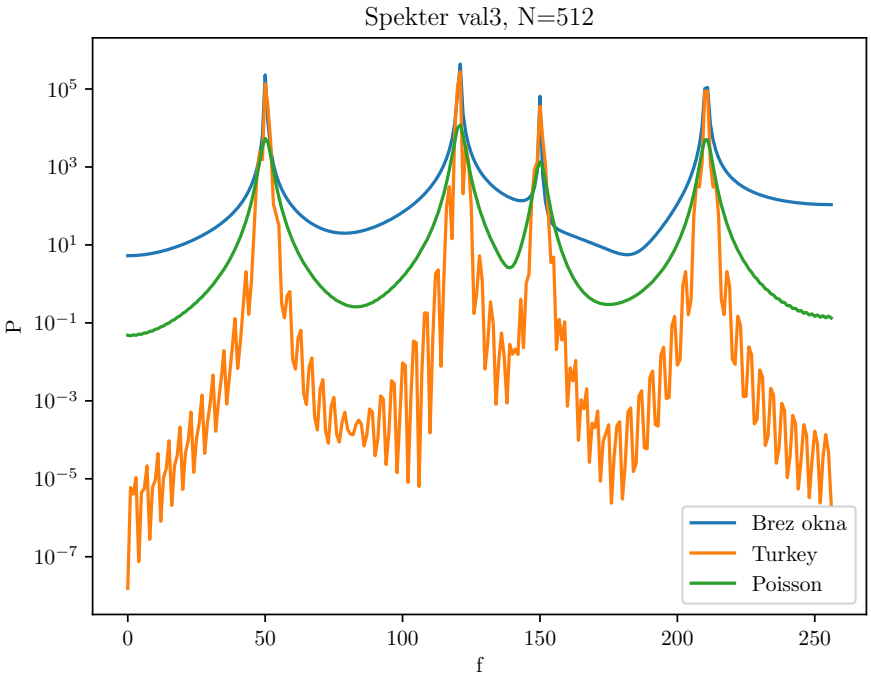
$$SRS(n) = 1 - 1.93\ cos\left(\frac{2\pi n}{N-1}\right) + 1.29\ cos\left(\frac{4\pi n}{N-1}\right) - 0.388\ cos\left(\frac{6\pi n}{N-1}\right) + 0.028\ cos\left(\frac{8\pi n}{N-1}\right)$$



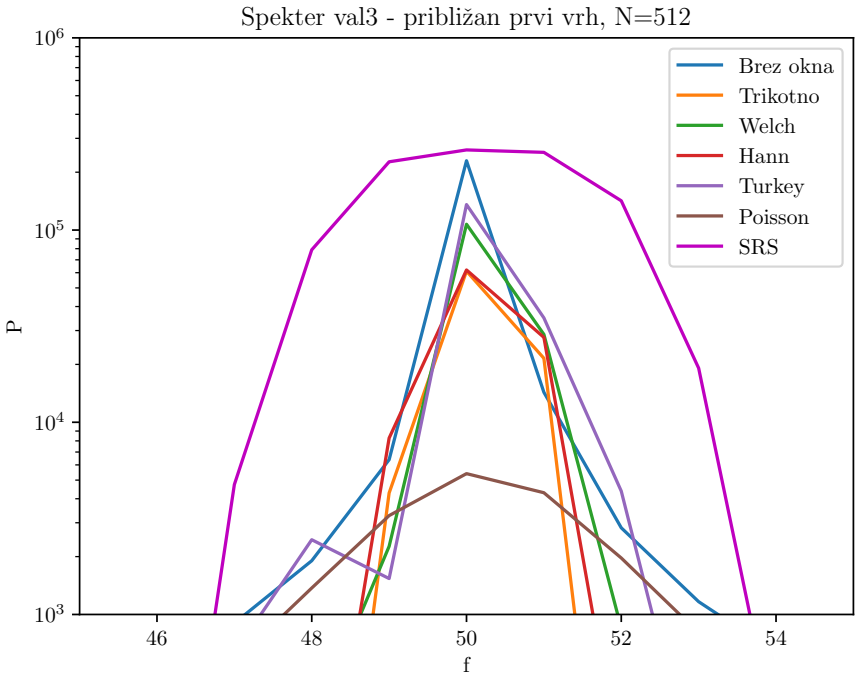
Okna, ki jih bom uporabil. Turško in Poissonovo sem dodal, ker sta se zdeli zanimivi saj predstavljata nekakšna limitna primera zelo škatlaskega in zelo ostrega okna. Okno SRS sem posebej dal na desno y os, ker je višje od ostalih.



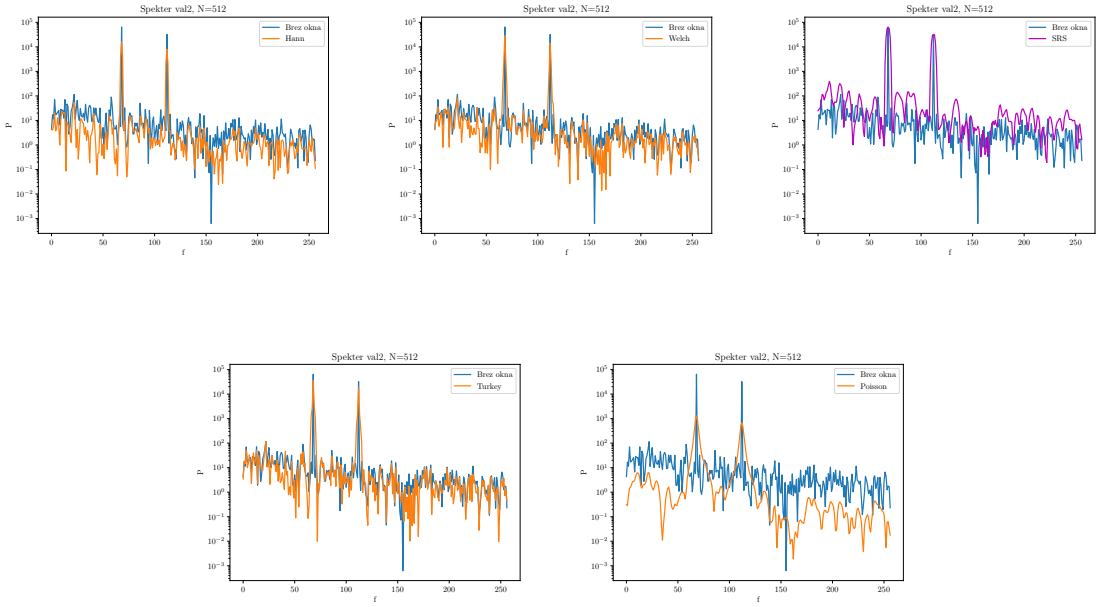
Odločil sem se, da okna stestiram raje na val3, ki ima lepše, osamljene vrhove. Brez okna je najslabše, frekvence izven vrhov sploh niso nizke, Welch to malo popravi, pri tem pa ohranja nekako širino vrha. Trikotno je podobno kot Welch, le da zraven nekako oscilira. Razlika je le na zadnjem vrhu, kjer Welch lepše osami vrh, a ima neke čudne skoke. Hann izgleda najbolje od vseh, še bolj potlači frekvence izven vrhov in zoži vrh. SRS jih potlači bolj kot Hann, a ima okoli vrhov zelo ozke hribčke.



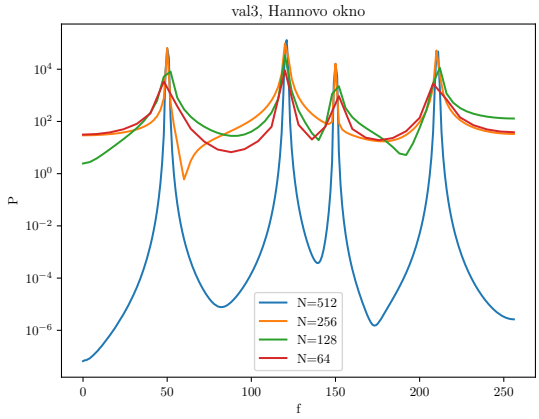
Turkey, katerega okno je najbolj škatlaste oblike potisne območja zunaj vrhov nekje med Welchom in Hannom, vrh ni preveč ozek, primerljiv je z Welchom. Poisson dela slabo, vrh celo opazno zniža, območja zunaj vrhov pa ne potlači preveč.

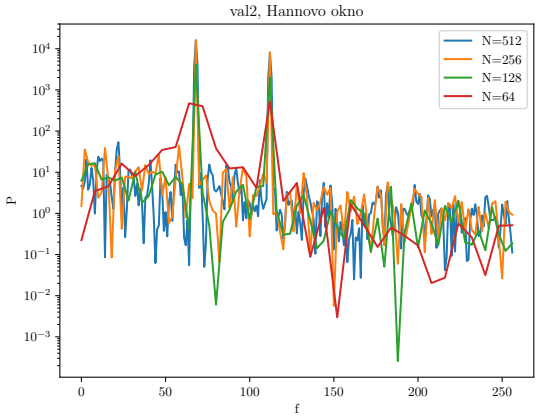


Približan prvi vrh spektra. Tukaj se lepo opazi, kako okna zožajo vrh. Jasno tudi vidimo še eno slabo lastnost SRS-a in sicer to, da vrh razširi. Prej je bil SRS mogoče še nekako primerljiv s Hannom, po tej sliki pa sem se odločil, da deluje Hann najbolje in ga bom od zdaj naprej uporabljal.

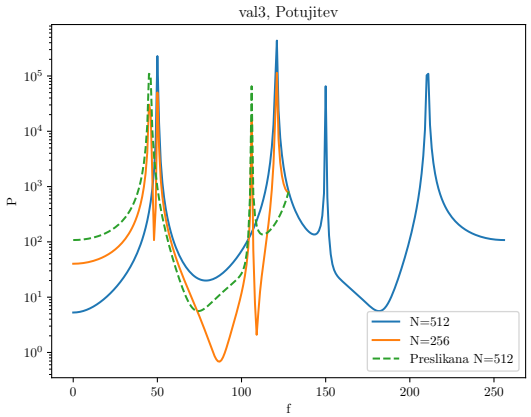


Slike so tukaj malce drugačne. Hann in Welch sta spet zelo blizu, a Welch malce prej razširi glavna vrhova. SRS in Poisson se slabo obneseta, še posebej SRS sploh ne potlači frekvenc zunaj vrhov. Turkey ni slab, a se vrh začne širiti še prej kot pri Welchu.





X os sem na grafih skaliral, da se vrhi prilegajo. Po pričakovanjih, da slabše vzorčenje slabši spekter.



Če namesto prvih 256 točk vzorčimo od vseh 512 vsako drugo, imamo nižjo frekvenco vzorčenja in posledično nižjo Nyquistovo frekvenco. Posledica je potujitev(aliasing), kar pomeni, da se del spektra nad našo Nyquistovo frekvenco preslika in pokvari spekter. Na sliki je z oranžno (potujen) signal z 256 točkami, z modro pa signal z 512 točkami. Črtkana zelena je desna stran modrega signala prezrcaljena čez sredino. Vidi se, kako to zrcaljenje ustvari nove vrhe pri oranžnem signalu.

2 Wienerjev filter

Tokrat imamo dan signal oblike

$$c(t) = (u(t) * r(t)) + n(t) = s(t) + n(t)$$

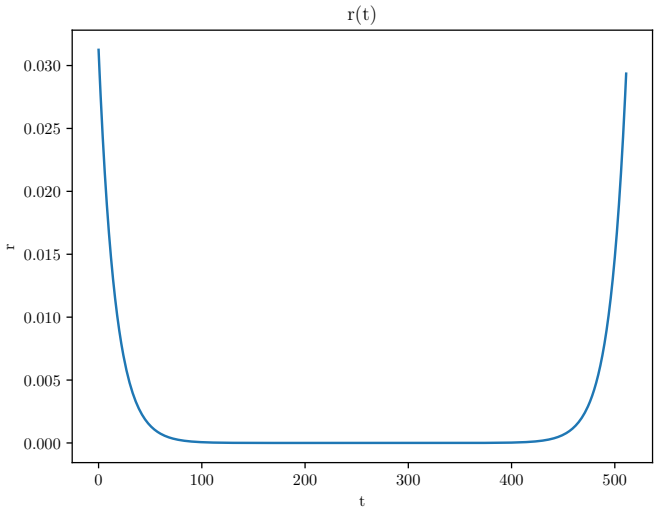
u je vhodni signal, katerega želimo rekonstruirati, r je znana prenosna funkcija, n pa neznan šum. * označuje konvolucijo, za katero vemo, da je v Fourierovem prostoru množenje.

Če šuma ni zlahka dobima prvoten signal:

$$\begin{aligned} &c(t) = u(t) * r(t) \\ &X(f) = \mathscr{F}(x(t)) \qquad\qquad C(f) = U(f)R(f) \\ &U(f) = C(f)/R(f) \\ &u(t) = \mathscr{F}^{-1}(C(f)/R(f)) \end{aligned}$$

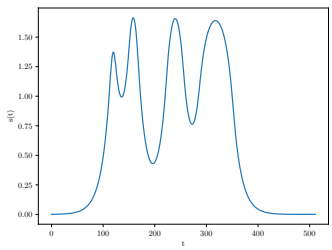
Prenosna funkcija je podana in je

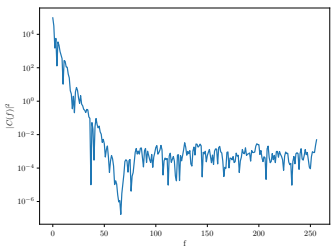
$$r(t) = \frac{1}{32} exp(-|t|/16)$$

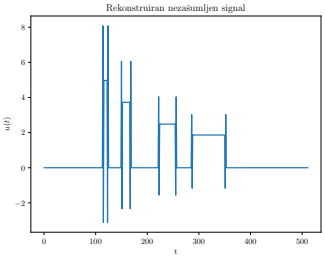


Funkcijo sem zashiftal za N/2, saj Fourierova transformacija predpostavlja periodičnost.

Naredimo to za signal0.dat za katerega vemo, da nima šuma:





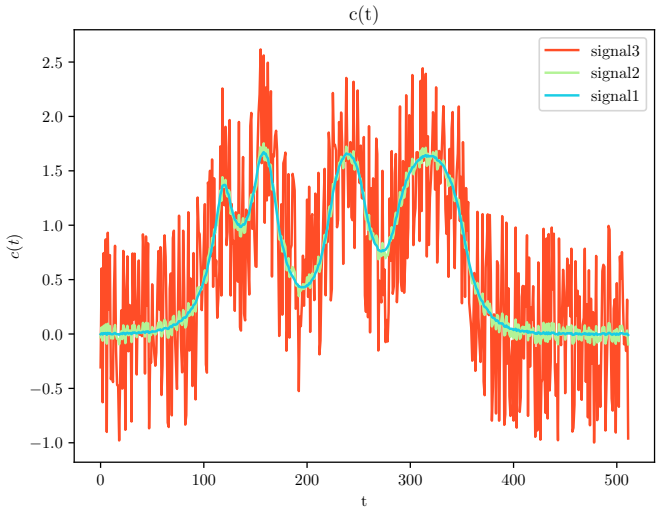


Na levi naš merjeni, že skonvoliran signal. Na sredini isti signal v Fourierovi sliki, tipično je vrednost nad neko frekvenco nič. Na desni reskonstruiran prvotni signal. Vidimo, da je to le reskaliran merjeni signal.

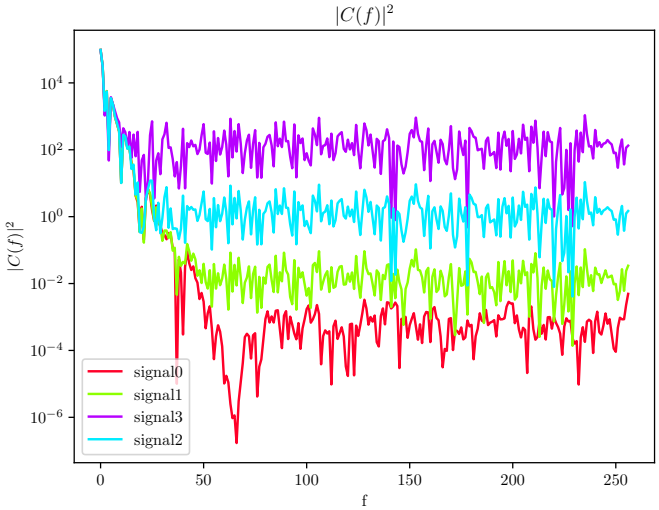
Pri ostalih signalih imamo primešan šum. Tega se bomo skušali znebiti z Wienerjevim filtrom:

$$u(t) = \mathscr{F}^{-1}\left(C(f)/R(f)\Phi(f)\right)$$
$$\Phi(f) = \frac{|S(f)|^2}{|S(f)|^2 + |N(f)|^2}$$

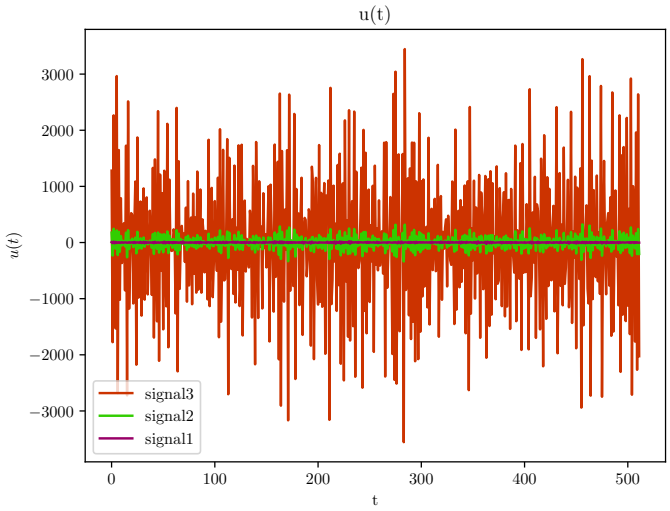
Šum N(f) bo treba nekako določiti, predpostavil bom, da je to le konstantno ozadje, ki ga bom poskusil določiti iz frekvenčne slike $c(f)$.



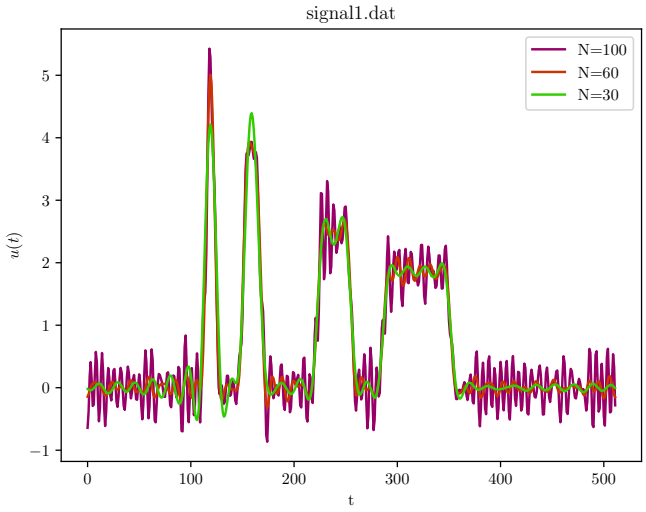
Naši zašumljeni signali.



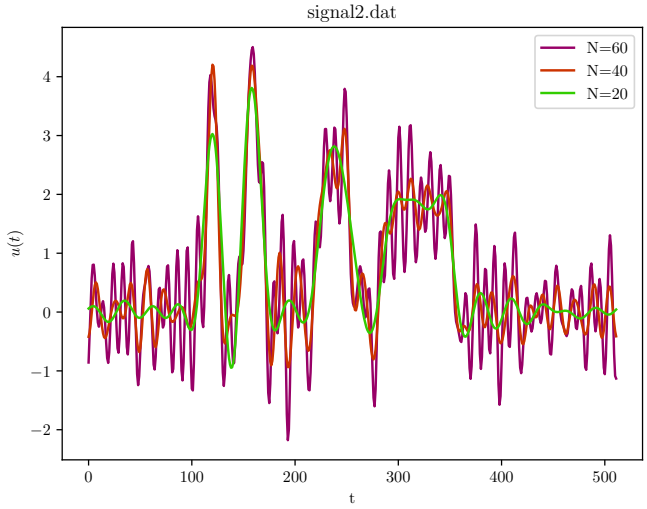
Res zgleda, da bi šum lahko bil konstanten. Tukaj bi lahko goljufali in samo odšteli nezašumljen signal za natačno določitev šuma, a se bom raje delal, da nezašumljenega signala ne poznam in skušal na roke določiti koliko je offset od ničle pri visokih frekvencah.



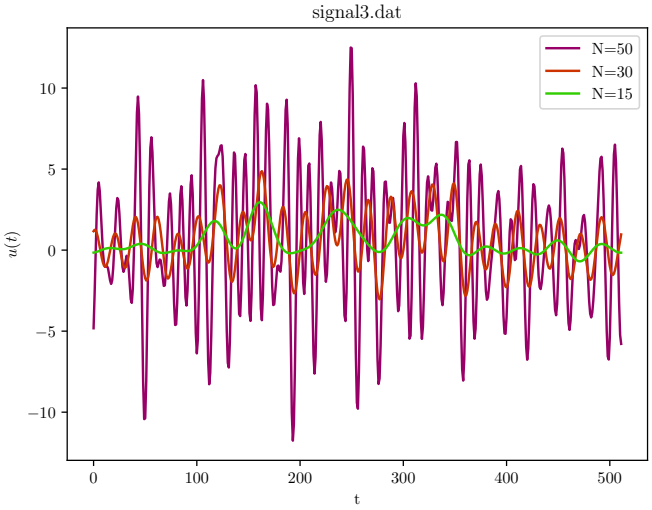
Če se ravnam po prej opisanem postopku pride kar nekaj. Še nekaj bo treba nardit



Če poleg prejšnjega dam še funkcijo $\Phi(f)$ na 0 na tistem frekvenčnem območju kjer predvidevamo, da gre le za šum, dobimo pravilen spekter, ki je seveda vseeno manj natančen za bolj zašumljen signal. Oblika signala je odvisna od tega od katere frekvence naprej dam Φ na 0. To sem označil na zgornjem grafu z N. Če je N manjši imamo v signalu prisotne le manjše frekvence, kar nam manj natančno definira robove "kvadratastega" signala, a nam po drugi strani signal manj oscilira med "kvadrati".



Pri bolj zašumljenem signalu je bilo treba porezati več frekvenc.

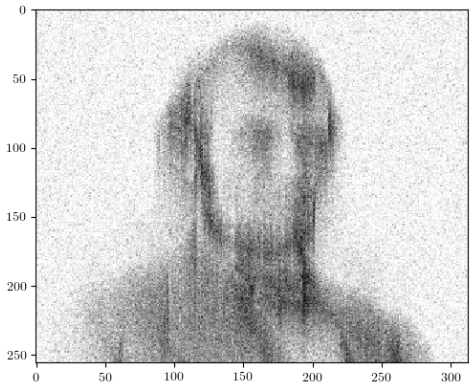


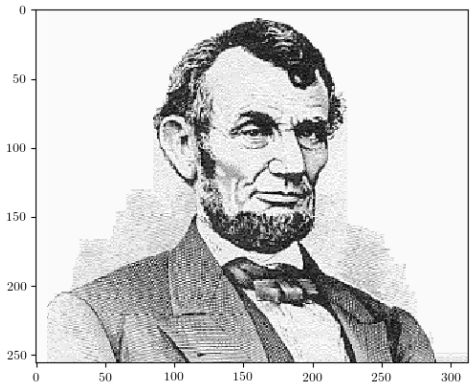
Tako močno zašumljen signal je še težje prepoznaven.

3 Čiščenje slike

Zdaj bom Wienerjev filter uporabil za čiščenje slike Lincolna, odbrane po stolpcih. Prenosna funkcija za razmazanje slike je

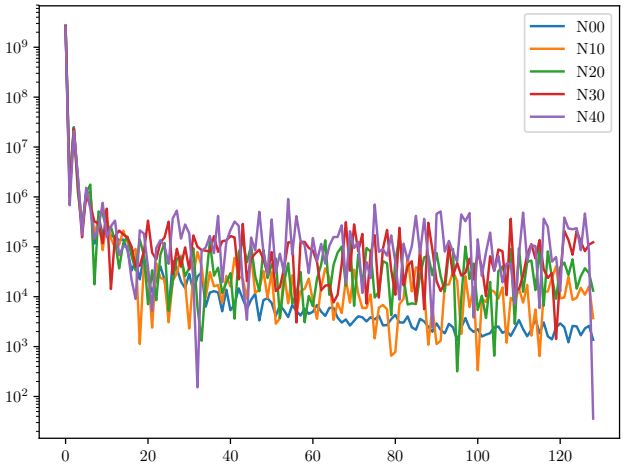
$$r(t) = \frac{1}{\tau} exp(-t/\tau)$$





Tako izgleda čiščenje slike brez šuma. Kar v redu. Inverzna fourierova transformacija je vrnila vrednosti piksol nad 255, katere sem moral ročno postaviti na 255.

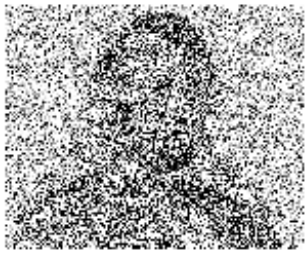
Naslednje slike, označene z N10,N20,N30,N40 so zašumljene. Predpostavil bom, da je šum enak čez vse stolpce in si ogledal spekter le za en stolpec:



Ni tako lepo kot pri prejšnji nalogi,a vseeno vidimo, da moč z frekvenco najprej pada potem pa se približno ustali. Za šum bom spet privzel kar konstantnega. N00 ustreza sliki brez šuma, ki ima tudi sicer tudi pri višjih frekvencah neko moč, a ker je največja moč pri nižjih bom frekvence na tistem območju vseeno kar porezal in upal, da kot prej pride pregleden rezultat.

N10, Original

N10, Brez filtra



N10, N=100

N10, N=70

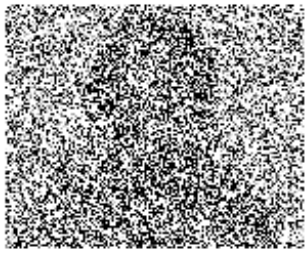
N,10 N=50

N10, N=30

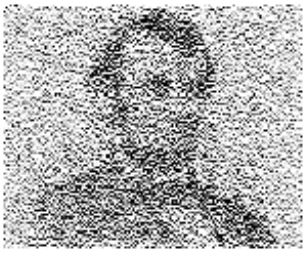
Brez filtra se vidi očiten šum čez sliko. S filtrom pa opazno bolje. N spet označuje od kje naprej porežemo frekvence. Pri N=100 ostane še nekaj šuma, pri nižjem N pa ni več opazen, a se to pozna na kvaliteti slike

N20, Original

N20, Brez filtra



N20, N=100



N20, N=70

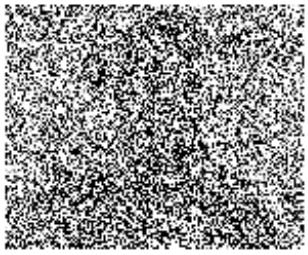
N20, N=50

N20, N=30

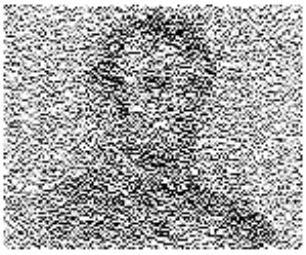
Vidi se, da je tukaj šum veliko večji. Frekvence moramo še prej porezati, da pride slika brez šuma.

N30, Original

N30, Brez filtra



N30, N=100



N30, N=70

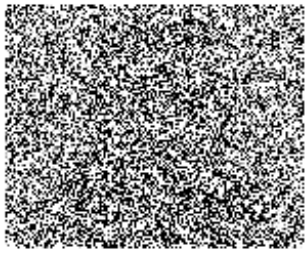
N30, N=50

N30, N=30

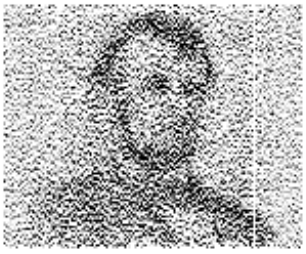
N=20 je zdaj še komaj dobro.

N40, Original

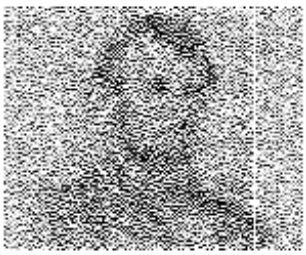
N40, Brez filtra



N40, N=50



N40, N=30



N40, N=20

N40, N=10

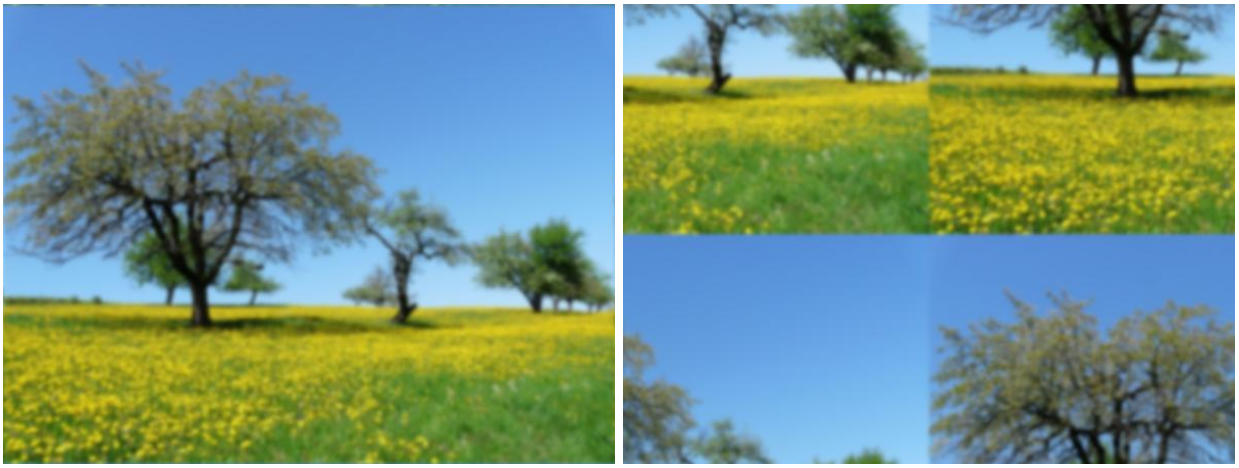
Šum je odpravljen šele pri N=10, kjer je kvaliteta slika zelo slaba. Na očiščeni sliki je opažen tudi en stolpec, ki se je nekako izgubil.

4 Gaussian Blur

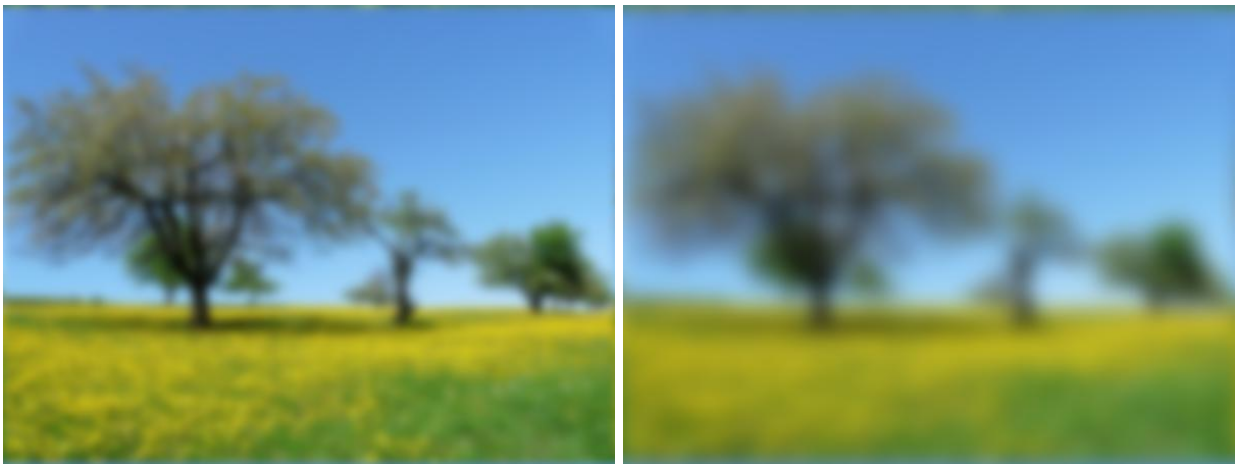
Hočem sprogramirati še program, ki mi bo na sliki izvajal tako imenovan "Gaussian blur" filter. To je nekakšen učinek, ki sliko zamegli. To se naredi tako, da funkcijo najprej po stolpcih nato pa po vrsticah skonvoliram z gaussovo funkcijo.

Poskusil bom zamegliti naslednjo sliko, kjer je vsak piksel prestavljen z 3 integerji med 0 in 255(RGB). Sliko bom transformiral s FFT in v Fourierovem prostoru pomnožil z Gaussom. Za obdelavo slik uporabim Pythonovo knjižnico PIL.

Original slika, katero bom obdelal.



Slika, skonvolirana z Gaussovim šumom z standardno deviacijo 2 piksla. Razlika med slikama je, da sem na levi Gaussovko centriral na polovico širine slike, na desni pa sem je dal polovico na vsak levi in desni rob slike(periodičnost)



Še sigma 5 in 10. Dobro deluje!

Tukaj sem skonvoliral le rdečo komponento pikslov(sigma 5 in 10), ostale pa pustil pri miru. Dobimo zanimiv efekt.

Sigma 10 še pri konvoluciji le zelene oz. modre komponente