

Дигитални аудиосистеми

Бранислав Геразов

Факултет за електротехника и информациски технологии
Универзитет Св. Кирил и Методиј во Скопје, Македонија

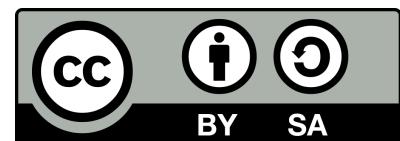
Дигитални аудиосистеми ~ Предавања и вежби v0.75

©Copyright Бранислав Геразов, 25 мај 2017 г.

Скрипта од предавањата и вежбите по предметот Дигитални аудиосистеми
Институт за електроника

Факултет за електротехника и информациски технологии
Универзитет Св. Кирил и Методиј во Скопје, Македонија

This work is licensed under a Creative Commons Attribution-
ShareAlike 4.0 International. [https://creativecommons.org/
licenses/by-sa/4.0/](https://creativecommons.org/licenses/by-sa/4.0/)



Содржина

1 Вовед во дигитален звук	5
1.1 Предности на дигиталниот звук	5
2 Дигитализација на звукот	8
2.1 Земање одбиоци или семплирање	8
2.2 Дискретизација по амплитуда или квантизација	13
2.3 Формати на дигитално аудио	19
3 Вовед во работа со звук во Python	21
3.1 Слободен и отворен софтвер во научните истражувања	21
3.2 Употреба на Python за истражување	22
3.3 Поставување на Линукс средината	23
3.4 Основи на Python	24
3.5 Основи на NumPy и Matplotlib	24
3.6 Воведни материјали за Python, NumPy, Matplotlib, SciPy	25
3.7 Вчитување на звук во Python	25
3.8 Скратување на звукот и префранање во моно	26
3.9 Прикажување на аудиосигналот	26
3.10 Преслушување на аудиосигналот	27
3.11 Менување на амплитудата на аудиосигналот	28
3.12 Нормализација на аудиосигналот	30
3.13 Генерирање на звук во Python	31
4 Фреквенциски спектар на звучните сигнали	33
4.1 Основи на Фурьеовата анализа	33
4.2 Анализа на спектарот на звучните сигнали	38
4.3 Фурьеова трансформација на временски отсекоци STFT	43
5 Филтри	49
5.1 Основи на дигиталните филтри	49
5.2 Дизајн на FIR филтер	53
5.3 Споредба на карактеристиките на FIR и IIR филтри	55
5.4 Еквализација	55
5.5 Филтри непропусни на фреквенција – notch филтри	58
5.6 Дигитални аудиоэффекти базирани на филтри	59
6 Процесирање на аудиосигналите базирано на доцнење	60
6.1 Основи на процесирање со задоцнување	60
6.2 Echo	61
6.3 Реверберација	64
6.4 Други дигитални ефекти базирани на доцнење	64
7 Компресија на аудиосигналите	65

7.1	MPEG-1 ниво III	66
7.2	Компресија на говор	69
8	Процесирање на аудиосигналите базирано на линеарна предикција	71
8.1	Анализа и синтеза на глас со линеарна предикција	72

Поглавје 1

Вовед во дигитален звук

Како што преодот кон обработка и зачувување на звукот во електричен домен донел своевидна револуција во квалитетот на снимениот звук, така преодот на звукот од електричен во **дигитален домен** уште повеќе го издигнал нивото на поимањето на звукот во техниката и секојдневието. Во извесна смисла станува збор за втора **револуција** и тоа: во начинот на зачувување на звукот, во неговата обработка и анализа, како и во неговата масовна дисеминација. Навистина, дигиталниот запис со својата недвосмисленост овозможува безгрешно преснимување на звучните записи.

Дигиталните алатки се речиси семоќни кога станува збор за обработка на звучните записи. Од просто засилување и слабеење на сигналот, што се сведува на просто множење на нумеричкиот запис на сигналот со одреден коефициент, до комплексни аудио ефекти, синтеза на нови звучни облици, издвојување на мелодиска линија, отстранување на шумот итн. – обработката на звукот е ограничена само од човековата визија за она што сака од звукот да го добие. Уште повеќе, ако се има в' предвид дека компјутерите и микроконтролерите се главните платформи за обработка на дигиталното аудио, тогаш може да се опремата потребна за тоа е лесно достапна и масовно раширен, а не привилегија на професионалците.

1.1 Предности на дигиталниот звук

Дигиталното аудио во однос на аналогниот запис на звукот носи низа на придобивки. Во продолжение ќе се осврнеме на главните.

1.1.1 Поширок динамички опсег

Дигиталното аудио кодирано со 16–битна резолуција теориски нуди однос сигнал шум од 96 dB, споредено со 80^{te} dB динамички опсег кај најдобрите аналогни системи. Ова ниво е уште поголемо ако го зголемиме бројот на битови со кои го кодираме звукот. Ваков динамички опсег во реалноста не е навистина неопходен. Тој доаѓа во игра само во исклучителни случаи кога во рамките на едно музичко дело еден симфониски оркестар ја искористува целосната динамика која може да ја даде. На пример тоа е случај кога имаме релативно тивки мелодии на флејта па потоа громогласни изливи на целиот оркестар. Тогаш менувањето на силината на влезниот сигнал со потенциометар не е дозволено.

1.1.2 Зголемена отпорност на шум

Дигиталните системи се имуни на загадувањето на звучниот сигнал со шум. Додека кај аналогните системи шумот предизвикан од електромагнетните пречки, како и термичкиот

шум се акумулираат во звучниот сигнал долж неговото движење низ електронските кола; во дигиталните системи таква акумулација на шум нема. Професионалната дигитална аудио опрема работи со 24–битно кодирање на звукот, што соодветствува со однос сигнал шум од 144 dB, а единствениот присутен шум е оној кој ќе влезе во аудио системот пред степенот за дигитализација.

1.1.3 Усовршено и олеснето умножување:

Дигиталниот аудио запис може да се пресними од еден дигитален уред на друг без било каква загуба во квалитетот, додека кај аналогните записи секогаш имаме загуби во квалитетот на записот па и додавање на нов шум при преснимувањето. Така, дури и кај најдобрите аналогни системи постои загуба од околу 3 dB во односот сигнал шум при правењето на копија на некој запис. Ако имаме синцир на преснимувања копија од копија од копија, квалитетот на конечната снимка ќе биде намален за сумата на штетни влијанија внесени од секој од уредите искористени во синцирот. Со други зборови квалитетот на аналогната снимка е условен од должината на синцирот, како и од квалитетот на уредите кои учествуваат во него. Дигиталното преснимување е неосетливо на обата параметри.

Уште повеќе, умножувањето на дигиталните аудиозаписи е олеснето поради зголемената брзина со која можат да се направат копиите. Кај аналогното преснимување, поради самата карактеристика на аналогните носачи на звук, речиси секогаш мора да се прави во реално време. Така за преснимување на грамофонска плоча на аудио касета во најдобар случај мора да пројде онолку време колку што трае самиот звучен запис. Истото важи и кај преснимувањето од касета на касета. Некои системи нудат двократно скратување на ова време со зголемена брзина на движење на магнетните ленти при преснимувањето. Сепак во двата случаја за преснимување 60–минутен запис скоро секогаш се потребни 60 минути преснимување. Од друга страна, за да преснимиме 80–минутно аудио CD на хард дискот на компјутерот ни требаат 5 минути, а за понатамошно умножување на записот во рамките на хард дискот по копија ни се потребни само 15 секунди!¹

1.1.4 Механизми за корекција на грешка

Поради претставувањето на звукот ефективно со низи од единици и нули, погодно е со помош на ефикасни методи за кодирање и внесување на редунданса, да се осигури интегритетот на дигиталниот аудио запис. Повеќето дигитални носачи на звук како на пример CD–то и DAT касетите имаат вградени механизми за корекција на грешка. Ако површината на дискот физички е оштетена, читачот автоматски ја употребува сета останата информација да ги реконструира изгубените податоци.

1.1.5 Зголемена трајност на дигиталниот запис

Оптичките носачи на звук како CD–ата се многу поотпорни на стареење од магнетните носачи на звук. Кај нив не постои саморазмагнетизација како кај магнетните ленти а непостоењето на физички контакт меѓу механизмот за читање и самиот диск ги оневозможува оштетувањата при преслушувањето на записот. Најголеми штетни влијанија врз долготрајноста на оптичките медиуми на запис се радијацијата, влажноста, како и температурните влијанија. Сепак, производителите на CD–R дискови рекламираат трајност од 75 години при соодветно складирање, наспроти 30^{te} години максимални за магнетната лента. Кај златните и платинестите CD–а оваа граница оди и до 100, односно 200 години соодветно!

¹Точните вредности зависат од неколку параметри на компјутерскиот хардвер, но се згодни за добивање на перспектива на временскиот сооднос.

1.1.6 Неограничени можности за обработка и монтирање

При работа со звукот за првпат тој може визуелно да се претстави и директно да се работи со таа негова визуелизација. Во компјутерските системи звукот е зачуван како временска низа од амплитудни вредности која лесно може да се прикаже на екраните при обработката. Тоа овозможува бескрајна прецизност при сечењето и монтирањето на аудио материјалот. Уште повеќе како низа од податоци звукот е достапен за математичка анализа и обработка со помош на софтверски алатки, без потреба од софистициран хардвер.

Поглавје 2

Дигитализација на звукот

Под **дигитализација** се подразбира процесот на префлување на еден аналоген процес, односно сигнал, во дигитален домен. Кај звукот работиме со неговата претстава во аналоген електричен домен добиена како што веќе кажавме, со електроакустички претворувач – микрофон. Така добиениот електричен сигнал е од аналоген карактер, односно тој е континуиран во време и е со континуирана распределба на амплитудите. Амплитудата на сигналот може да биде еднаква на било кое напонско помеѓу двете максимални нивоа на сигналот. Исто така, при премин од една конкретна вредност на неговата амплитуда во друга, тој поминува низ бесконечен број на состојби (во време и амплитуда) на патот меѓу нив.

Првиот чекор во дигитализацијата е **дискретизирање во време** на овој аналоген електричен сигнал, уште наречено и **семплирање**¹. Дискретизацијата во време се прави со периодично земање одбироци од дадениот аналоген сигнал во одредени временски интервали. Добиениот временски дискретен сигнал е во вид на диракови импулси, но тој суштите има континуирано распределени амплитуди. За да се претстави амплитудата на секој од одбироците со конечен број на битови, потребно е заокружување на нивните амплитуди до известни конкретни вредности од множеството вредности кои можат да се запишат. Ова е вториот чекор во дигитализацијата – **дискретизација по амплитуда** или **квантизација**. Во скlop на квантизацијата се прави и последниот чекор од процесот – **кодирањето** на амплитудните вредности со низи од единици нули, со што конечно го добиваме звукот претставен во дигитален домен. Овие три чекори се спроведуваат во соодветни електронски кола наречени **аналогно–дигитални (AD) конвертори**. Обратниот процес, односно реконструирањето на аналогниот сигнал од неговата дигитална претстава се врши соодветно со уредите наречени **дигитално–анalogни (DA) конвертори**.

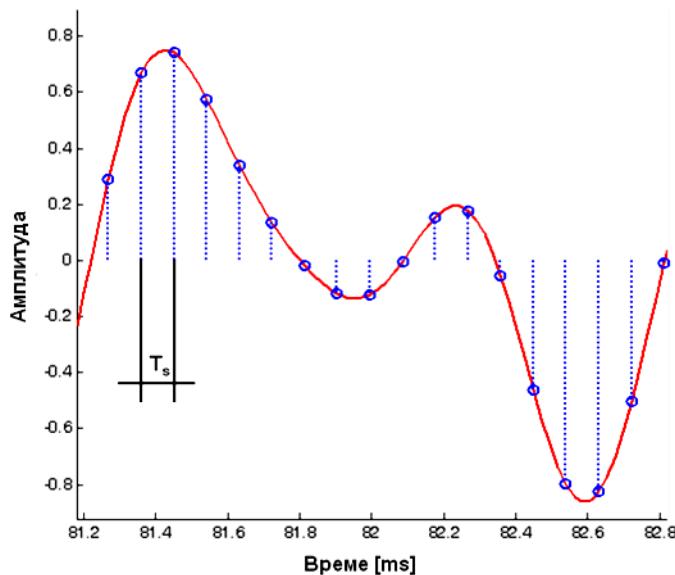
2.1 Земање одбироци или семплирање

Земањето одбироци претставува запомнување на вредностите на аналогниот влезен сигнал согласно со ритамот одреден од фреквенцијата на земање одбироци F_s , односно во временски интервали T_s дефинирани како:

$$T_s = \frac{1}{F_s}. \quad (2.1)$$

Процесот на земање одбироци е прикажан на Сл.2.1. Ова се прави со помош на соодветни **Sample/Hold (SH)** електронски кола, а може идеално да се претстави како множење на аналогниот сигнал со низа на диракови импулси. Главната идеа е временската информација

¹Важно е да се напомене дека во музиката под поимот семплирање се подразбира земањето на отсекоци од готови музички траки или звуци од музички инструменти и нивната употреба за креирање на ново музичко дело.



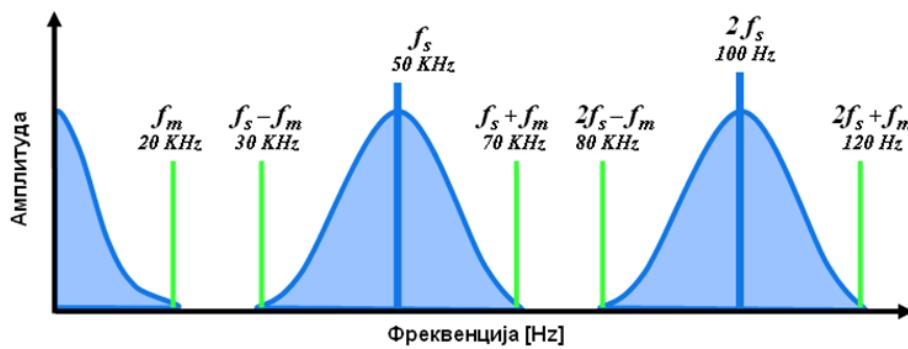
Сл. 2.1: Земање од битови со фреквенција F_s , односно период T_s .

за сигналот содржана во земените од битови да биде доволна, за да може од нив потоа тој верно да се реконструира. На прв поглед ова изгледа невозможно – како може нешто што трае континуирано во време да се претстави со конечна низа на вредности? Што станува со информацијата за сигналот помеѓу од битовите? Сепак, математички е докажано дека ова е возможно. Станува збор за познатата **теорема за земање од битови**, до која дошле повеќе автори, и тоа: Хери Найквист, Клод Шенон, Едмунд Витакер и Владимир Котельников, кој за првпат ја применува во телекомуникациите во 1933^{та} година. Теоремата за земање од битови гарантира дека за веродостојно да се зачува целосната информација содржана во еден аналоген сигнал со максимална фреквенција на неговиот спектар F_m , доволно е од него да земаме земање од битови со фреквенција $F_s \geq 2F_m$. Минималната фреквенција на земање од битови $F_s = 2F_m$ се нарекува **фреквенција на Найквист**.

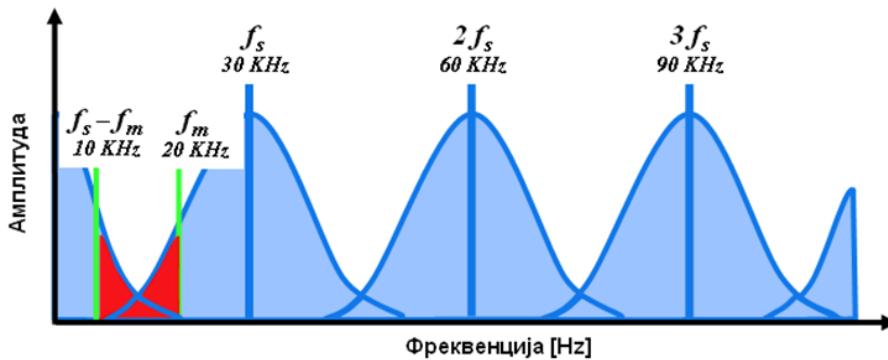
§ Дополнително. Аудио–CD стандардот употребува фреквенција на земање примероци од 44,1 kHz што соодветствува на максимална фреквенција на спектарот на сигналот од 22 kHz. Ова е поголем опсег од слушниот опсег на човекот. Тој е избран поради тоа што кога се воведувал аудио CD стандардот во 1980^{та} единствениот систем кој овозможувал фреквенциски опсег на записот доволно голем за да се зачува звукот во дигитален формат бил VHS системот за видео касети. Тогаш постоеле ИКМ адаптери кои аналогното аудио го дигитизираше а потоа повторно го претворале во аналоген видео сигнал кој се носел во видеото за снимање. Тие можеле да запишат по 3 примероци од секој аудио канал во една хоризонтална видео линија. Бидејќи PAL системот има 294 линии и фреквенција на работа од 50 Hz, следува дека брзината на земање примероци може да изнесува $294 \cdot 50 \cdot 3 = 44.100$ примероци/секунда.

Спектарот на дискретизираниот сигнал е ист со овој на оригиналниот, со тоа што сега постојат **копии** на овој спектар центрирани на целобройни мултипи од фреквенцијата на земање на битови F_s како на Сл. 2.2. На пример, ако звукот кој е ограничен на 20 kHz го дискретизираме со F_s од 50 kHz, тогаш слики од оригиналниот спектар ќе се наоѓаат во интервалите од 30 – 70 kHz, 80 – 120 kHz итн.

Бидејќи овие копии на спектарот на сигналот не смеат да се преклопуваат потребна е фреквенција на земање битови поголема од $2F_m$, односно од Найквистовата брзина. Од друга страна бидејќи сигналите скоро никогаш немаат строго ограничен спектар, мора сигналот да се претфильтрира и да се сведен на фреквенциски опсег со максимална фреквенција F_m .



Сл. 2.2: Спектар на дискретизираниот сигнал.



Сл. 2.3: Преклопување на спектрите при ниска фреквенција на земање на одбороци.

Кога сигналот би имал спектрални компоненти над F_m , или пак кога би го дискретизирале со фреквенција $F_s < F_m$, сликите на оригиналниот спектар би се преклопувале. **Преклопувањето** на спектрите, во англиската литература познато како **aliasing**², прикажано е на Сл. 2.3. Тоа внесува изобличувања во оригиналниот спектар на сигналот – оној во опсегот од 0 до F_m . Поради тоа, реконструкцијата на аналогниот сигнал нема да соодветствува на оној пред дискретизацијата.

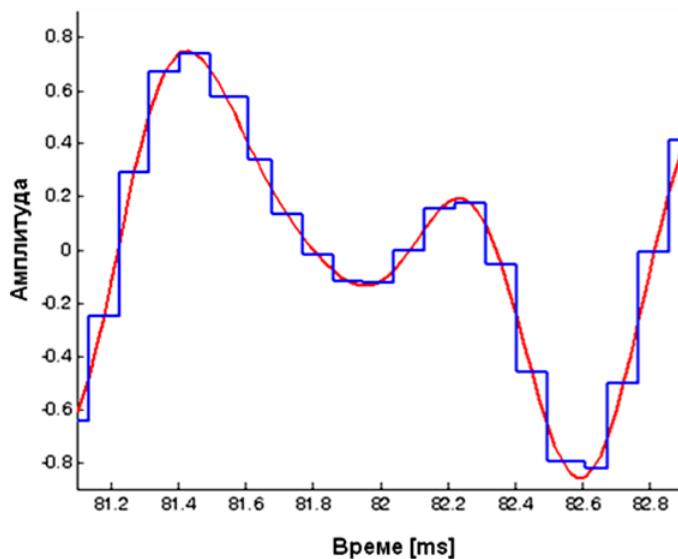
Реконструирањето на аналогниот сигнал од неговите одбороци се врши со употреба на **ниско–пропусен (НФ) филтер** со гранична фреквенција F_m . Тој во фреквенциски домен има задача да ги отстрани сликите од спектарот на сигналот лоцирани околу мултиплите на фреквенцијата на земање одбороци F_s , по што ќе остане само оригиналниот спектар на сигналот. Во временски домен тоа се сведува на измазнување на дискретниот сигнал, прикажано на Сл. 2.4.

2.1.1 Фреквенции на семплирање во дигиталното аудио

Брзини со кои се врши земањето одбороци во дигиталното аудио се:

- **8.000 Hz** – се употребува во телефонската комуникација и е доволна за пренесување на говор во однос на зачувување на неговата разбираливост,
- **11.025 Hz** и **22.050 Hz** – четвртина и половина од фреквенцијата на земање на одбороци во аудио CD стандардот 44.100 Hz, се употребува за зачувување на дигитално аудио со понизок квалитет,
- **32.000 Hz** – се употребува во miniDV стандардот за дигитално видео кој го користат дигиталните видео камери, во Digital Audio Tape (DAT) стандардот во модовите за

²Од зборот alias што значи лажно име, односно лажно претставување.



Сл. 2.4: Реконструкција на аналогниот од дискретизираниот сигнал со НФ филтрирање.

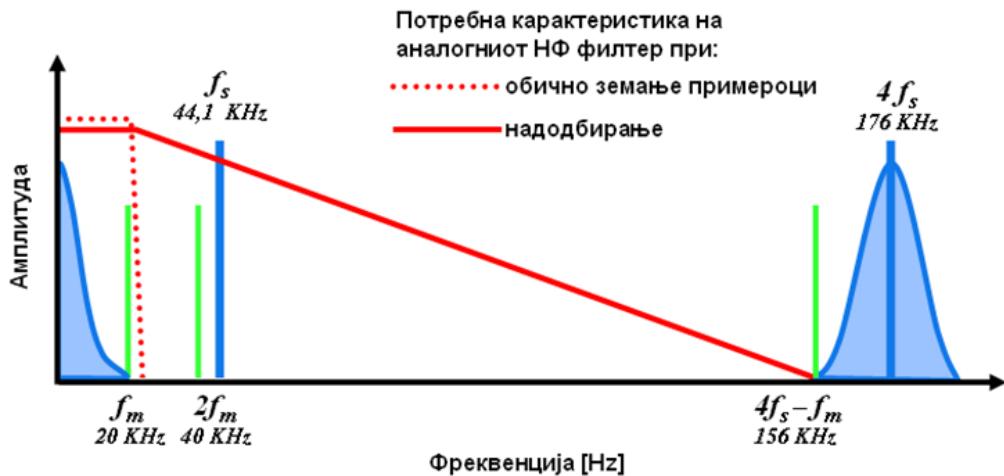
зголемено траење на касетата (long play mode), како и Германското дигитално сателитско радио Digitales Satelitenradio,

- **44.100 Hz** – фреквенција на земање одбирачи кај аудио CD-то, наследена од ИКМ адаптерите, исто така најчесто се употребува кај MPEG кодираното аудио (како кај VCD, SVCD, MP3),
- **48.000 Hz** – дигитално аудио во употреба кај miniDV, дигиталната Телевизија, DVD, DAT, филмови и професионална аудио опрема,
- **96.000 Hz** или **192.000 Hz** – во употреба кај аудио DVD стандард, за аудиото во новата генерација на DVD-а со син ласер (Blu-ray Disc) и во HD-DVD (High-Definition DVD),
- **2,8224 MHz** – се употребува во SACD (Super Audio CD) стандардот развиен од Сони и Филипс, кој употребува 1-битна сигма-делта квантизација. Според некои истражување овој начин на дигитализација не нуди предности над стандардниот кој е сега во примена.

2.1.2 Надсемплирање

Директниот начин на кој може да се изведе земањето одбирачи со фреквенција од на пример 44,1 kHz, како кај аудио CD-то, е да се направи токму тоа – со еден AD конвертор да се одмери сигналот во кратки временски интервали со фреквенција од 44,1 kHz. Меѓутоа, ова бара влезниот аналоген НФ филтер во AD конверторот низ кој најпрвин минува сигналот, да е со многу стрма амплитудна карактеристика веднаш над 20th kHz. Тој ќе мора во многу краток фреквенциски интервал да обезбеди слабеење од повеќе од 80 dB, колку што изнесува потребниот однос сигнал/шум за Hi-Fi репродукција на звук. Но, бидејќи динамичкиот опсег кој го нуди дигиталниот аудио CD запис е над 90 dB, всушност слабеењето на филтерот треба да е уште поголема. Во практиката ова подразбира употреба на аналоген филтер од 8th или 10th ред што е скапо и непропорочливо решение, поради потребата за усогласеност на составните аналогни компоненти.

Постои уште еден поекономичен начин да се реализира земањето одбирачи, а тоа е со употребата на **надсемплирање**, во англиската терминологија познато како **oversampling**. Надсемплирањето претставува земање на одбирачи со фреквенција неколку пати поголема од Најквистовата. Со него се постигнува олеснување на условите кои влезниот НФ филтер



Сл. 2.5: Благиот наклон на влезниот НФ филтер овозможен со надодирањето во споредба со потребната карактеристика на филтерот при обично земање на одбороци.

треба да ги задоволи при обликувањето на аналогниот сигнал. Потребното ограничување на спектарот на сигналот, сега може да се изврши во дискретен (дигитален) домен каде НФ филтрирање на сигналот е многу поедноставно, а и поефтино. Уште една голема предност на дигиталното филтрирање над аналогното е строгата линеарност на фазната карактеристика која е гарантирана кај **FIR³ филтрите** со конечен импулсен одсив.

Потоа може да се отфрлат вишокот одбороци (одбороци), со што доаѓаме до целната фреквенција на земање одбороци.

Фреквенцијата со која се прави надсемплирање најчесто се движи од $4\times$, $8\times$, па сè до $32\times$. Најчесто тоа се прави со 4 или 8-кратно поголема фреквенција. Во SACD стандардот се употребува фреквенција на надодирање од 2,8224 MHz што соодветствува на $64\times$ надсемплирање.

Пример 1. За да ги илустрираме придобивките од надсемплирањето ќе земеме $4\times$ надсемплирање во однос на аудио-CD стандардот, тогаш фреквенцијата на надсемплирање f'_s ќе биде 4 пати поголема од номиналната (целната) F_s од 44,1 kHz и ќе изнесува:

$$f'_s = 4F_s = 4 \cdot 44,1 = 176,4 \text{ kHz}. \quad (2.2)$$

Во овој случај за да нема преклопување на спектрите кај дискретниот сигнал, потребно е спектарот на сигналот да не содржи значителни компоненти над:

$$f'_m = \frac{f'_s}{2} = \frac{176,4}{2} = 88,2 \text{ kHz}. \quad (2.3)$$

Така аналогниот филтер може да биде со поблаг пад во карактеристиката од претходно и сепак да ги задоволи условите за висок квалитет на записот. Падот може да започне нешто над 20 и да трае сè до 88 kHz, каде треба да го достигне слабеењето од 80 dB. Уште повеќе, нас не нè интересира верна репродукција на целиот опсег од 88,2 kHz кога корисниот аудио сигнал ни се наоѓа до 20 kHz. Значи, фреквенцијата по која филтерот треба да го постигне бараното слабеење се поместува надесно до $176,4 - 20 = 156,4$ kHz, како што е прикажано на Сл. 2.5. Следува дека сега преодниот фреквенциски опсег на филтерот изнесува 120 kHz споредено со 2 kHz во случајот кога не се користи надсемплирање. Ова во голема мера ја поедноставува изградбата на аналогниот филтер.

³Finite Impulse Response.

По надсемплирањето, сигналот ги следи останатите два чекора на дигитализацијата (квантизација и кодирање) со што од него се добива дигитален сигнал. При тоа поради надодбирањето, овој дигитален сигнал е дигитална претстава на аналоген сигнал со спектрални компоненти од 0 до 88,2 kHz. Овој опсег нас не ни е од интерес, па затоа треба да го сведеме дигиталниот сигнал на фреквенција на земени одбироци од $F_s = 44,1$ kHz, која за нас беше целната фреквенција. Најпрвин за да го издвоиме корисниот спектар, дигиталниот сигнал го обработуваме со дигитален НФ филтер со стрмно отсекување на фреквенциите над 22 kHz, ова може лесно да се реализира во софтвер, односно во дигиталната интегрирана техника. Исто така филтерот може да има идеално линеарна фазна фреквенциска карактеристика, што е тешко да се направи во аналоген домен, но и не е критично во аудио апликациите поради спомнатата фазна неосетливост на човековото уво.

Потоа ја намалуваме фреквенцијата на земените одбироци преку процесот на **скастрување**, односно . Ова наједноставно се прави со отфрлање на 3 од секои 4 одбироци со што добиваме верна претстава на аналогниот звучен сигнал со спектар до 22 kHz во дигитален домен. Во стварноста не се отфрлаат по 3 одбирока од секои 4 туку самиот дигитален филтер ја пресметува вредноста на еден излезен одбирок на секои 4 влезни со нивно усреднување.

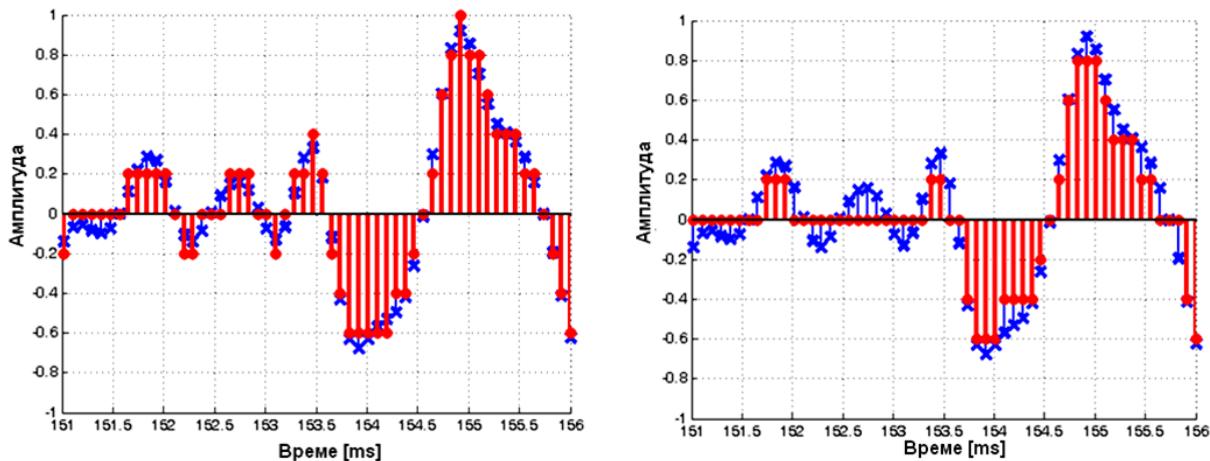
§ Дополнително. Процесот на надсемплирање наоѓа примена и во реконструкцијата на аналогниот сигнал од неговата дигитална претстава. Каде DA конверторите надодбирањето се изведува со додавање на екстра нулти одбироци на секој реален одбирок. Тука тоа се нарекува **up-sampling**. Притоа нивната амплитуда најчесто се интерполира помеѓу вредностите на реалните одбироци или пак математички се моделира како што е најповолно за DA конверторот. Овој процес е фундаментално различен од надсемплирањето при дискретизацијата на влезниот сигнал. Имено тој не внесува додатна информација во дигиталниот сигнал. Сепак, како и претходно, надсемплирањето ја олеснува изградбата на излезниот аналоген НФ филтер во DA конверторот, преку зголемувањето на минималната фреквенција на првата слика на звучниот спектар.

2.1.3 Цитер

Цитерот претставува грешка во временските моменти во кои се прави семплирањето поради варијациите во временската база на уредот за дигитализација. Така A/D конверторот со фреквенција на земање одбироци од 44,1 kHz, не секогаш ги зема одбироците точно секој 44100^{ти} дел од секундата. Тие може да бидат земени малку порано или покасно со што снимената вредност на влезниот сигнал не сосема соодветствува на вредноста која тој би ја имал во предвидениот временски момент. Ефектите на цитерот на временска база стануваат понагласени при присуство на високи фреквенции, како и на големи амплитуди кај влезниот сигнал.

2.2 Дискретизација по амплитуда или квантација

За теоремата за земање одбироци да важи целосно, амплитудата на секој одбирок мора да е еднаква со онаа на влезниот сигнал во тој временски момент. Меѓутоа поради тоа што секој одбирок во дигиталниот запис мора да се запише (кодира) со ограничен број на битови, наречени **дигитален збор**, следува дека и бројот на амплитуди, односно множеството амплитуди кои можат да се запишат е конечно. Така, ако бројот на битови во дигиталниот збор е N , тогаш, вкупниот број на амплитуди кои со него можат да се запишат е 2^N . Поради тоа континуираните вредности на влезните амплитуди мора да се заокружат или да се пресликаат во ним најблиските им вредности во рамките на оваа множества. Тој процес се нарекува



Сл. 2.6: Квантација со заокружување (лево) и со отсекување (десно).

квантација, а електронскиот уред со кој таа се изведува **квантизер**⁴. Бројот на битови кој квантизерот може да го генерира се нарекува уште и негова **резолуција**.

Квантацијата се одвива на тој начин што квантизерот врши споредба на амплитудата на влезниот сигнал со дискретните нивоа кои тој може внатрешно да ги генерира, наречени **квантизациски нивоа**. Разликата помеѓу две соседни квантизациски нивоа се нарекува **чекор на квантација** и се означува со q . Споредбата тече во неколку чекора низ кои квантизерот ја „лови“ амплитудата на влезниот сигнал, од најгрубата негова проценка, која соодветствува на половина од неговиот референтен напон V_{REF} , а се кодира со битот со најголема тежина – MSB⁵, до најголемата прецизност која тој може да ја постигне, која е $V_{REF}/2^N$ и се кодира со битот со најмала тежина LSB⁶. Процесот е воден од внатрешната логика на квантизерот. Важно е брzinата со која се доаѓа до конечниот коден збор да е поголема од брзината со која доаѓаат одбироците од влезниот сигнал.

Постојат два главни типа на квантација кои се разликуваат според начинот на кој го ловат влезниот сигнал. Тие се **квантација со заокружување** и **квантација со отсекување**. Кај првата излезот на квантизерот го дава квантационото ниво најблиско до амплитудата на влезниот сигнал, додека кај втората тоа е квантационото ниво веднаш под амплитудата на влезниот сигнал. Типот на квантација зависи од електронската реализација и механизмот на работа на квантизерот. Разликите помеѓу двата типа се илустрирани на Сл. 2.6.

2.2.1 Квантација на дигитално аудио

При дигитализација на аудиото се употребуваат следните резолуции на квантација:

- **1 бит** – кај супер аудио-CD стандардот (SACD) кој користи надсемплирање и техники за обликување на шумот на квантација за постигнување на голем однос сигнал-шум и при 1-битна резолуција.
- **4 бита** – ретко се употребува при линеарна импулсно кодна модулација (ИКМ), а во употреба кај адаптивната квантација.
- **8 бита** – во употреба во телефонските системи за кодирање на говорот, стандардна резолуција на старите звучни карти во ерата на флопи-дискетите.

⁴Влезниот аналоген филтер, земачот на примероци, квантизерот и на крај кодерот заедно творат еден AD конвертор.

⁵Most Significant Bit.

⁶Least Significant Bit.

- **12 бита** – во употреба во 1980^{тите} во дигиталните уреди за ефекти употребувани во музичирањето, како за електричните гитари.
- **16 бита** – дел од аудио-CD стандардот, денес стандардна резолуција за повеќето уреди за дигитално аудио (на пример звучни картички).
- **20 бита** – вградена во многу AD конвертори, во новата серија 20-битни ADAT повеќеканални машини и кај некои звучни картички.
- **24 бита** – новиот стандард за висока дефиниција, имплементиран во нови професионални уреди за дигитално аудио, новите DAT машини и во аудио DVD стандардот.
- **32 и 64 бита** – не се употребува за снимање на звук, туку за неговата дигитална обработка и спектрална анализа со софтверските алатки; служи за зголемување на точноста во пресметките, особено значајно кога тие би вовеле дисторзија во 16-битно запишан звук.

2.2.2 Кодирање

Под **кодирање** се подразбира начинот на кој дискретната амплитуда на квантизерот ќе биде запишана во битови и тоа е вградено во самата негова изведба. Наједноставниот начин на кодирање на амплитудите на влезниот сигнал е со нивно директно претворање во бинарни (кодни, дигитални) зборови. Притоа паровите амплитуда-коден збор се **линеарно распоредени**, со најниската амплитуда запишана со најмалиот дигитален збор (сите 0), а највисоката амплитуда со најголемиот дигитален збор (сите 1). Малку посложена е **нерамномерната квантизација**, односно кодирање, кај која амплитудите со најголема веројатност на појавување пофино се квантанизираат од оние кои поретко се појавуваат. На пример, за крајно високите амплитуди се отстапени помалку кодни зборови отколку за амплитудите поблиску до нулата. Исто така постојат и методи на квантизација кои ја кодираат **разликата** помеѓу две последователни амплитуди на влезниот сигнал. Тие се употребуваат во апликации каде големината на дигиталниот проток е критичен параметар.

2.2.3 Шум на квантација

Грешката при квантација може да ја сметаме и како шум додаден на некоја инаку идеална амплитудна вредност. Гледано од тој аспект, таа се нарекува и **шум на квантација**. Под услови:

1. влезниот сигнал да може да се разгледува како **слушаен процес** и
2. тој да има **многу поголема амплитудна динамика** од чекорот на квантација q , што е еквивалентно на тоа чекорот на квантација да е доволно мал,

шумот на квантација може да го моделираме како **бел шум**, односно тој ќе биде **слушаен и некорелиран** со влезниот сигнал, со средна вредност 0 во случај на квантација со заокружување, односно $q/2$ кога се врши квантација со отсекување. Во овие услови, моќноста на шумот на квантација ќе биде рамномерно распределена долж целиот спектар и со константа амплитуда наречена **ниво на шумот на квантација**. Во праксата дава услови се исполнети речиси секогаш, особено за сигнали кои имаат мал коефициент на корелација, како што се говорот и музиката, како и кога квантизерот е со доволно голема резолуција. Условите не се исполнети само во многу специјални случаи како што се константни сигнали, простопериодични сигнали синхронизирани со фреквенцијата на дискретизација и многу слаби сигнали.

Односот меѓу просечната моќност на аналогниот сигнал и нивото на шумот на квантација се нарекува **однос сигнал-шум (SNR)**⁷ на дигиталниот сигнал. Неговата вредност може да се добие со помош на равенството:

⁷Signal-to-Noise Ratio.

Табела 2.1: Варијанса на Гаусовиот процес употребен за моделирање на неколку звучни сигнали.

Вид на звучен сигнал	σ
Чалгиска музика	0,150
Изворна музика	0,236
Пеење	0,326
Говор	0,223

$$SNR = 10 \log \frac{\sigma_x^2}{\sigma_q^2}, \quad (2.4)$$

каде σ_x е варијансата на влезниот случаен аналоген сигнал, а σ_q е варијансата на шумот на квантизација. Ако влезниот сигнал $x[n]$ ги задоволува гореспоменатите два услови тогаш густината на веројатност на грешка е рамномерно распределена во интервалот од $-q/2$ до $q/2$ па за σ_q добиваме:

$$\sigma_q^2 = \frac{q^2}{12}. \quad (2.5)$$

Ако земеме дека $V_{REF} = 1$, тогаш $q = \frac{1}{2^N}$, па добиваме:

$$\sigma_q^2 = \frac{1}{12 \cdot 2^{2N}}, \quad (2.6)$$

па (2.4) станува:

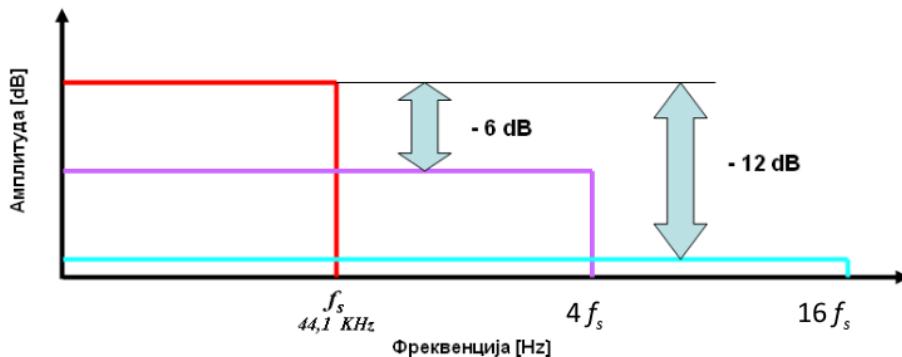
$$SNR = 10 \log \left(\frac{\sigma_x^2}{\frac{1}{12 \cdot 2^{2N}}} \right) = 10 \log(12 \cdot 2^{2N} \sigma_x^2). \quad (2.7)$$

При пресметување на равенството (2.7) можеме влезниот сигнал да го моделираме како Гаусов, при што е важно тој да не ја надмине границата на квантизаторот, којашто земавме дека е 1. Ако варијансата на Гаусовиот процес $\sigma_x < 0,25$, тогаш веројатноста да се случи пречекорување е занемарлива, т.е. $< 0,0001$, што речиси секогаш е исполнето при моделирањето на звукот како Гаусов процес. Неколку вредности на варијансата на Гаусовиот процес добиен со моделирање на различни типови звук се дадени во Табела 2.1.

Конечниот израз за односот сигнал-шум во однос на должината на кодниот збор N кој се добива со земање на σ_x е:

$$SNR = 6,02n - 1,25 \simeq 6n [\text{dB}]. \quad (2.8)$$

Во праксата речиси секогаш се користи упростената форма, поради поврзаноста на константниот член со варијансата на звучниот сигнал, како и поради неговата едноставност. Како што гледаме, односот сигнал-шум расте за 6 dB со секој додатен бит во должината на кодниот збор. Според тоа за односот сигнал-шум во аудио-CD стандардот, со 16-битна резолуција, добиваме вредност од 96 dB (94,75 dB). За резолуција од 24 бита по одбирок пак, добиваме однос сигнал/шум од 144 dB, што е еквивалентен на слушниот опсег на човекот.



Сл. 2.7: Намалувањето на нивото на шумот со употреба на надсемплирање.

2.2.4 Квантација при надсемплирање

Ако влезниот аналоген сигнал е случаен процес, па шумот на квантација е бел и некорелиран, тој ќе биде рамномерно распределен долж целото фреквенциско подрачје кое го зафаќа сигналот. Што се случува ако направиме $4\times$ надсемплирање – односно ако AD конверторот семплира со фреквенција од $4\times 44,1\text{ kHz}$? Во овој случај шумот од квантација ќе биде распределен на 4 пати поширок фреквенциски опсег, па соодветно и амплитудата на неговата спектрална густина на моќност ќе биде 4 пати помала. Тоа соодветствува на ниво на шум помало за $10 \log^{1/4} = -6\text{ dB}$. Со други зборови $4\times$ надсемплирање има ист ефект врз односот сигнал-шум како и зголемувањето на должината на дигиталниот збор за 1 бит. Слично, $16\times$ надсемплирање соодветствува на зголемена резолуција на AD конверторот од 2 бита, како што е прикажано на Сл. 2.7.⁸

Ова може да се каже и на следниов начин: 16–битен AD конвертор со брзина на земање одбирачи од $44,1\text{ kHz}$ нуди ист однос сигнал-шум како 15–битен AD конвертор со $2\times$ надодбирање. Ако одиме понатаму по оваа логика стигнуваме до поедноставени AD конвертори кои со помалку битови по одбирок но со многу поголема фреквенција на одбирање нудат перформанси исти со оние на посложните 16 или 24 битни AD конвертори. Крајната точка во оваа дискусија е реализација на квалитетни AD конвертори кои работат со 1 бит по одбирок, како кај супер аудио-CD (SACD) стандардот⁹.

§ Дополнително. Надодбирањето одбирачи нуди подробности и во обратната насока, односно кај DA конверторите. Тука повторно шумот се прераспределува на поширок фреквенциски опсег па можно е отфрлање на дел од битовите по одбирок, а притоа задржување на истиот однос сигнал-шум.

2.2.5 Дитер

Претходната дискусија за нивото на шум на квантација важи во случај кога шумот на квантација е потполно некорелиран со влезниот сигнал. Во праксата, колку повеќе амплитудата на влезниот сигнал се приближува кон амплитудата означена со најмалку значајниот бит, толку повеќе се зголемува корелираноста на шумот на квантација со самиот сигнал. Како последица шумот на квантација не може веќе да се моделира како некорелиран бел шум. Неговата корелираност значи дека шумот на квантација ќе има периодична природа

⁸Ова може да се протолкува како поништување на шумот при усреднување на екстра земените примероци. Сличен процес се користи во астрономската фотографија каде поради слабата осветленост, шумот од сензорот е многу голем, па се земаат низа од фотографии од објектот и се усреднуваат во една во која шумот е видливо намален.

⁹SACD користи и техники за обликување на шумот дискутирани во Поглавјето 2.2.6 за уште поголеми придобивки во SNR.

па соодветно ќе внесе сопствени хармоници во спектарот на сигналот. Увото ги толкува овие изобличувања како непријатна дисторзија.

Дитерот¹⁰ е процес со кој може да се елиминира појавата на дисторзија во сигналот под дејство на шумот од квантизација. Тој е воведен од Лоренс Робертс¹¹, еден од основачите на интернетот, во својата магистерска теза на MIT во 1961. Во него на влезниот сигнал пред дигитализацијата му се додава мало ниво на аналоген шум. Случајниот карактер на шумот внесува случајност и во однесувањето на квантизерот при ниски влезни нивоа на сигналот – ефективно внесувајќи несигурност во неговата работа. На тој начин грешката од квантизација се декорелира од сигналот. Ова доаѓа од таму што сега, излезните нивоа од квантизерот веќе не зависат само од влезниот сигнал туку и од некорелираниот случаен процес внесен како шум.

Концептот зад треперењето може интуитивно да изгледа нелогичен, но принципот на кој функционира е многу едноставен. Треперењето се базира врз одредено специфично однесување на човековото уво. Увото е помалку чувствително на широкопојасен шум со рамномерно ниво, отколку на хармониски изобличувања во спектарот на сигналот кои се јавуваат при корелираност на шумот со сигналот. Бидејќи увото може да открие звук во опсегот на средните фреквенции 10 до 12 dB под нивото на шумот, со треперење може да се постигне 18-битна резолуција кај еден 16-битен запис. Имено, при правилен избор на сигналот со кој се изведува треперењето се овозможува да се запишат сигнали и 110 dB под максималното ниво, иако 16-битната резолуција теориски нуди динамички опсег од 96 dB. Треперењето е во редовна употреба при намалувањето на резолуцијата на еден дигитален звучен запис, како што е редовно случај при мастерирањето. Во тој случај, материјалот снимен во студиото со професионална опрема со 24-битна резолуција, треба да се прилагоди за дистрибуција на носач на звук – CD со 16 бита. Со внесувањето на одредено ниво на шум, дитер, пред конверзијата се овозможува зачувување на дел од вишокот информација која ја содржи оригиналниот запис. Ако ова не се направи, тогаш простото отфрлање на информацијата зачувана во 24-битна резолуција со исфлање на долните 8 битови се нарекува **скастрување**. Ова неминовно ќе доведе до појава на корелиран шум на квантизација како дисторзија во конечниот аудиозапис.

§ Дополнително. Дитерот бил употребен прв пат во II Светска војна, кога забележале дека механичките компјутери за пресметување на траекторијата на бомбите при бомбардирањето работеле подобро во авионите отколку надвор од нив. Се покажало дека причината за ова е постојаното тресење на авионот кое овозможило полесно движење на деловите од овие компјутери кои биле замастени и слепени. Подоцна минијатурни мотори – вибратори наречени dither-и се вградувале во сите механички компјутери.

2.2.6 Обликување на шумот

Дитерот додава шум со рамна спектрална распределба на моќноста во аналогниот сигнал, но увото не е еднакво осетливо во целиот слушен опсег како што може да се види од изофонските криви прикажани на Сл. 2.8. Изофонските криви ја прикажуваат субјективната јачината на звукот и физичкото ниво на звучниот притисок. Први ги измериле Флечер и Мунсон (1933), па потоа и Робинсон и Дадсон (1956), за сега да бидат стандардизирани со ISO стандардот 226:2003. Тие ја отсликуваат фреквенциски нелинеарната карактеристика на сетилото за слух.¹²

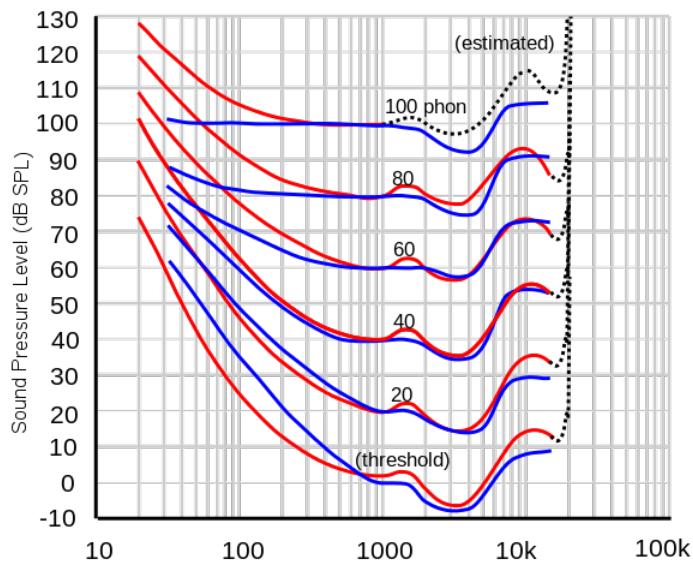
Со техниката на **обликување на шумот** може да ја менуваме фреквенциска распределба на дитерот, што ни овозможува поголем негов дел да го распределиме надвор од фреквенцискиот опсег во кој увото е најосетливо. Притоа вкупната моќност на шумот останува иста. Со

¹⁰Македонскиот термин за дитер би бил „треперење“, а следи од адаптација на англиското „to dither“, чија потекло е од старо-англискиот збор „didderen“ кој значи тресење (како од студ).

¹¹Wikipedia: Lawrence Roberts https://en.wikipedia.org/wiki/Lawrence_Roberts_%28scientist%29

¹²Повеќе за нив може да чуете во делот **Психоакустика** во предметот **Електроакустика**.

¹³„Lindos4“ by Lindosland <https://commons.wikimedia.org/wiki/File:Lindos4.svg#/media/File:Lindos4.svg>



Сл. 2.8: Криви на еднаква чујност, или изофонски криви, измерени според Флечер и Мунсон (сино) и стандардизирани според ISO стандардот (црвено).¹³

помош на овој пристап може во еден 16-битен сигнал да се постигне однос сигнал-шум на средни фреквенции дури од 150 dB. Обликувањето на шумот е особено згодно при употреба на надсемплирање, каде шумот може да се потисне надвор од слушното подрачје – над 20^{0т} kHz. Како и треперењето и обликувањето на шумот се употребува најмногу при конверзијата на дигиталниот сигнал од поголема во помала резолуција.

Пример 2. Каде 64× надсемплирање во SACD стандардот, фреквенцијата на земање одбирачи изнесува 2,8 MHz, па шумот може да се распределат до 1,4 MHz што овозможува однос сигнал-шум од 100 dB при кодирање со 1 бит, што според (2.8) би требало да овозможи динамички опсег од само 6 dB.

2.3 Формати на дигитално аудио

По земањето на одбирачи, квантизацијата и кодирањето ја добиваме трансформацијата на звукот во дигитален домен. Методот со кој е извршена дигитализацијата го одредува и начинот на кој звукот ќе биде запишан со низа од 1 и 0, односно го одредува **форматот на дигиталното аудио**. Основен формат на дигитално аудио е аудиото кодирано со **импулсно кодна модулација (ИКМ)**¹⁴. ИКМ претставува најквалитетниот, најсигурниот и наједноставниот начин на дигитално запишување на звучните аналогни сигнали и се користи кога не ни е пречка големината на битскиот проток генериран при дигитализација. Поради релативната едноставност и сигурност во однос на другите типови на квантизација и кодирање, какви што се на пр. адаптивната ИКМ и линеарно–предиктивното кодирање, ИКМ е најраспространетиот формат за кодирање на високо квалитетно аудио.

ИКМ кодираниот дигитален звучен сигнал на компјутерот се зачувува во вид на датотека, додека пак кај аудио-CD-то тој е запишан во вид на постојан проток, дефиниран според Црвената Книга на аудио-CD стандардот. Разликата е во тоа што на датотеките содржат воведен дел „header“ во кој се зачувани информации за должината на датотеката и параметрите на дигиталното аудио, како фреквенцијата на земање одбирачи и резолуцијата на записот, додека кај аудио-CD-то овие информации се излишни. Formatot во кој е зачувано ИКМ аудиото во компјутерските системи зависи од оперативниот систем кој го користи компјутерот. Windows оперативните системи го

¹⁴Pulse Code Modulation (PCM).

користат **WAV**¹⁵ форматот, развиен од страна на Микрософт за Интел базираните компјутерски системи во 1991, врз база на општиот RIFF (Resource Interchange File Format) дефиниран зачувување на разни видови на податоци. Компјутерите со Macintosh платформи го употребуваат **AIFF**¹⁶ форматот на датотеки, развиен од Apple во 1988. Двата паралелни стандарда се базираат на IFF форматот за зачувување на податоци воведен од Electronic Arts во 1985, со клучна разлика во редоследот на запишување на дигиталните податоци. WAV форматот најпрвин ги запишува 8^{te} бита со најмала тежина, па потоа тие со поголема, што во информатиката се нарекува „little-endian“ тип на запис. Од друга страна, AIFF форматот го прави тоа обратно, со запишување на байтите од најзначајниот кон најмалку значајниот, познато како „big-endian“.

§ Дополнително. Во WAV форматот се користат два начини на запишување на негативните одбироци во аудио сигналот, во зависност од бројот на битови по одбирок. Ако станува збор за аудио квантизирано со 8 бита по одбирок тогаш амплитудните вредностите се зачуваат без назнака за знакот `uint8`. Најмалата вредност што може да биде запишана `0x00`, соодветствува на најнегативната вредност на сигналот, додека најголемата вредност `0xFF`, соодветствува на неговата најпозитивна вредност. Кај 16-битната квантизација, негативните вредности на сигналот се зачуваат со употреба на двоен комплемент. Со тоа првиот бит од 16^{te} го дефинира знакот.

Дигиталното аудио често се користи во негова компримирана форма, со што се постигнува намалување на побарувањата за меморија и дигитален проток при неговото зачувување и пренос. За таа цел се развиени низа на техники за компримирање кои ќе бидат разгледани во Поглавјето 7.

¹⁵Waveform audio format.

¹⁶Audio Interchange File Format.

Поглавје 3

Вовед во работа со звук во Python

Со преминот на звукот во **дигитален домен**, се отвораат вратите кон примена на различните техники на дигиталната обработка на сигналите за негова обработка. Во оваа вежба ќе се запознаеме со основните принципи на работата со звукот во дигитален домен со употреба на програмскиот јазик **Python**.

3.1 Слободен и отворен софтвер во научните истражувања

Еден од најкористените за оваа намена е програмскиот пакет **Матлаб**¹. Матлаб, или „Matrix Laboratory”, нуди големи можности за дигитална обработка на разнородни податоци и сигнали. Тој може да се употреби за развој на алгоритми во најразлични области од инженерската практика, меѓу кои во дигиталната обработка на звук, слика и видео.

Денес сè повеќе научници своето истражување го засноваат на платформи базирани на **отворен софтвероопшто**². Ова пред сè се должи на филозофијата на отворениот, а особено **слободниот софтвер**³ основан од **Ричард Сталман**, како и поширокото **движение за отвореност**⁴, а тоа е заедништво во создавањето и напредувањето на технологијата и човештвото. Од практичен аспект, отворениот софтвер има низа предности над затворениот софтвер и тоа:

- **нултата цена** – поради основната премиса на давање на изворниот код, со цел да се овозможи неговиот развој од заедницата, отворениот софтвер е *de facto* и бесплатен софтвер. Така, повеќето производители на слободниот софтвер живеат од донацији.⁵
- **безбедноста** – поради достапноста на изворниот код, не постои начин производителот на софтерверот да прави нешто скриено од вас, а секој спорен дел од кодот е подложен на промена од заедницата. Кај затворениот софтвер тоа не е случај.⁶

¹MATLAB®, The MathWorks, Inc., Natick, Massachusetts, United States. <http://www.mathworks.com/products/matlab/>

²Wikipedia. *Open-source software*. https://en.wikipedia.org/wiki/Open-source_software

³Wikipedia. *Free software movement*. https://en.wikipedia.org/wiki/Free_software_movement

⁴Wikipedia. *Open-source*. https://en.wikipedia.org/wiki/Open_source

⁵Цената на комерцијалниот софтвер може да достигне многу големи суми, особено ако се работи за негова примена надвор од академската заедница. На пример, лиценца за основната инсталација на Матлаб, т.н. core, чини \$2.650 за индивидуални корисници, \$625 за академски корисници, а \$29 за студенти. Покрај неа, за вообичаена работа се потребни некои од специјализираните пакети, т.н. toolbox-и, на пр. Signal Processing Toolbox-от кој чини: \$1.250, \$250 и \$16 за трите типови на корисници, соодветно. Овие цени се наведени се моменталните дадени на <https://www.mathworks.com/pricing-licensing/index.html?intendeduse=comm&prodcode=ML>.

⁶На пример во Windows 10 производителот го задржува правото да ги чува вашите приватни податоци како што стои во изјавата за приватност: “Finally, we will access, disclose and preserve personal data, including your content (such as the content of your emails, other private communications or files in private folders), when we have a good faith belief that doing so is necessary ...” [1]

- **слободата од производителот** – како корисници на отворениот софтвер, вие не сте затворени во екосистемот на производителот. Истиот тој софтвер може да биде превземен од друга заедница на програмери и да продолжи неговото одржување и развој во друга насока.⁷
- **подobar квалитет** – при воспоставување на критична големина на заедницата околу еден отворен софтвер, развојот не може да се спореди со ресурсите кои ги поседува било која корпорација во светот. Така, развојот на **Линукс јадрото**⁸, кое во основата оперативниот систем **Линукс**⁹ и 600-те **GNU/Линукс дистрибуции**¹⁰, првично напишано од **Линус Торвалдс**, денес претставува најголемиот здружен проект во историјата на човештвото со околу 6000 активни развиваачи, над 20 милиони редови на код, и со проценета развојна вредност од над 2 милијарди евра.

3.2 Употреба на Python за истражување

Во духот на слободниот софтвер, а следејќи ги светските научни трендови, вежбите во предметот **Дигитални аудиосистеми** во целост ќе бидат изработени со слободен/отворен софтвер. За процесирањето на дигиталните аудио сигнали ќе биде искористен програмскиот јазик **Python**¹¹ и тоа неговата постара верзија **2.7**¹², заедно со библиотеките:

- **NumPy** – за работа со вектори и матрици,¹³
- **SciPy** – за дигитално процесирање на сигнали,¹⁴
- **Matplotlib** – за плотирање на 2Д графици.¹⁵

Освен овие постојат мноштво библиотеки за Python кои се користат во научните истражувања како на пример **Pandas**¹⁶ за статистички анализи, **Sympy**¹⁷ за симболичка математика, **SciKit-Learn**¹⁸ за машинско учење итн.

Како интерфејс кон Python ќе ја користиме интерактивната конзола **IPython**¹⁹ и научната развојна средина за Python **Spyder**²⁰.

Истите механизми се додадени во претходните верзии на Windows преку автоматските апдејти, но можат да се исклучват [2].

[1] Zach Epstein, Windows 10 is spying on almost everything you do – here's how to opt out, Jul 31, 2015.

<http://bgr.com/2015/07/31/windows-10-upgrade-spionage-how-to-opt-out/>

[2] Ashley Allen, How to Stop Windows 7 and 8 From Spying on You.

<http://www.eteknix.com/stop-windows-7-8-spying/>

⁷Една од стратешките определби на Apple е да ги затворат своите корисници во својот екосистем [3].

[3] Alonso Canada, Take A Lesson from Apple: A Strategy to Keep Customers in Your Ecosystem, Forbes, Nov 12, 2012.

<http://www.forbes.com/sites/jump/2012/11/12/take-a-lesson-from-apple-a-strategy-to-keep-customers-in-your-economy/#252f975343fa>

⁸Wikipedia. *Linux kernel*. https://en.wikipedia.org/wiki/Linux_kernel

⁹Wikipedia. *Linux*. <https://en.wikipedia.org/wiki/Linux>

¹⁰Wikipedia. *List of Linux distributions*. https://en.wikipedia.org/wiki/List_of_Linux_distributions

¹¹Python <https://www.python.org/>

¹²Ова се должи на некомплетно завршеното портирање на пакети развиени од научната заедница за работа со Python.

¹³NumPy <http://www.numpy.org/>

¹⁴SciPy <http://www.scipy.org/>

¹⁵Matplotlib <http://matplotlib.org/>

¹⁶Pandas <http://pandas.pydata.org/>

¹⁷Sympy <http://www.sympy.org/en/index.html>

¹⁸SciKit-Learn <http://scikit-learn.org/stable/>

¹⁹IPython Interactive Computing <http://ipython.org/>

²⁰Spyder – The Scientific PYthon Development EnviRonment <https://github.com/spyder-ide/spyder>

3.3 Поставување на Линукс средината

Иако користењето на **Python** не е врзано со Линукс оперативниот систем, овие вежби ќе се базираат на работа под Линукс. Доколку веќе немате Линукс, истиот се препорачува да го инсталirate паралелно на постоечкиот оперативен систем, во таканаречен *dual-boot* режим. Во најмала рака може да инсталите Линукс во виртуелна машина.²¹ Во Лабораторијата за дигитално процесирање на сигнали ќе работиме со **Ubuntu MATE**²² кој е базиран на **Ubuntu**²³, но наместо **Unity** го користи **MATE** десктоп интерфејсот базиран на **Gnome 2**.

За почеток треба во вашиот **home** фолдер да отворите нова папка со името на предметот. Тоа може да го направите преку фајл експлорерот, или преку **терминалот**²⁴, кој се стартува со комбинацијата **Ctrl+Alt+T**²⁵. Во терминалот напишете:

```
$ mkdir das
```

Следно, од веб страната на предметот Дигитални аудиосистеми превземете ја архивата со звучни сегменти кои ќе ги користиме во вежбите. Истата отпакувајте ја преку десен клик и аудиофајловите префрлете ги во ново направениот фолдер. Отпакувањето од **tar**-топката и префрањето може да го направите и преку терминал²⁶:

```
$ cd Downloads
$ tar -xf audio.tar
$ mv *.wav ~/das/
$ cd ~/das
```

За да може да ги слушнеме овие аудиозаписи треба да го инсталаме **SoX**²⁷ кој претставува моќна алатка за конверзија на аудиофајлови од еден формат во друг, но може да се искористи и за додавање на различни аудиоекфекти, како и снимање и преслушување на аудиофајлови. За инсталирање и надградба на софтверот и оперативниот систем во Линукс е одговорен менаџерот на пакети. Кај дистрибуциите од фамилијата на **Ubuntu** како менаџер се користи **apt-get**. Поради безбедносни причини во Линукс при секое менување на инсталираниот софтвер и системските фајлови мора да се повикаме на администраторски привилегии преку наредбата **sudo**²⁸:

```
$ sudo apt-get install sox
```

По што може да преслушаме некој од аудиофајловите:

```
$ play Solzi.wav
```

Solzi.wav:

```
File Size: 345k      Bit Rate: 714k
Encoding: Signed PCM
Channels: 1 @ 16-bit
Samplerate: 44100Hz
Replaygain: off
```

²¹ Добар преглед на популарноста на различните Линукс дистрибуции, како и повеќе информации за истите може да најдете на вебстраницата *Distrowatch*. <http://distrowatch.com/>

²² Ubuntu MATE. <https://ubuntu-mate.org/>

²³ Ubuntu. <http://www.ubuntu.com/>

²⁴ Еден добат сайт за да научите како да го користите Линукс терминалот е Linux Command. http://linuxcommand.org/lc3_learning_the_shell.php

²⁵ Ubuntu MATE доаѓа со прединсталiran терминал **Tilda** кој може да го повикате со F12.

²⁶ Терминалот овозможува автоматско комплетирање на кодот со употреба на Tab.

²⁷ SoX - Sound eXchange. <http://sox.sourceforge.net/>

²⁸ Кратенка од super user do.

Duration: 00:00:03.87

In:52.8% 00:00:02.04 [00:00:01.83] Out:90.1k [-====|=====] Clip:0

3.4 Основи на Python

За работа со Python може да ја искористиме стандардната конзола која доаѓа со неговата инсталацијата. Таа се повикува со:

```
$ python
```

```
Python 2.7.10 (default, Oct 14 2015, 16:09:02)
[GCC 5.2.1 20151010] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

За подобра работа со Python ќе ја инсталираме интерактивната конзола **IPython** која меѓу другото овозможува и автоматско комплетирање на кодот:

```
$ sudo apt-get install ipython
```

По инсталацијата може да ја повикаме со:

```
$ ipython
```

```
Python 2.7.10 (default, Oct 14 2015, 16:09:02)
Type "copyright", "credits" or "license" for more information.

IPython 2.3.0 -- An enhanced Interactive Python.
?           -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help        -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.
```

```
In [1]: print 'hello world'
```

hello world

...

За излегување од конзолата треба да притиснеме Ctrl+D.

3.5 Основи на NumPy и Matplotlib

Пред да се запознаеме со основите на работата со треба да ги инсталираме потребните модули:

```
$ sudo apt-get install python-numpy python-matplotlib python-scipy
```

Сега може да ги повикаме во IPython:

```
In [1]: import numpy as np
In [2]: from matplotlib import pyplot as plt
```

...

3.6 Воведни материјали за Python, NumPy, Matplotlib, SciPy

Добар вовед во програмскиот јазик **Python**, од аспект на неговата примена за истражување, може да се најде во книгата **Scipy Lecture Notes** (Varoquaux et al., 2015) која може во целост да се превземе од интернет. Оваа книга претставува отворен проект и во неа, благодарејќи на многуте автори и придонесувачи, се поместени основите за работа со Python, NumPy, SciPy, Matplotlib, SciKit-learn, SymPy, па дури и Cython.

Добат вовед во програмскиот јазик Python е и официјалниот туторијал кој може да се најде на неговата вебстраница²⁹. Уште една добра книга за основите на Python е книгата **Invent Your Own Computer Games with Python** (Sweigart, 2010) која исто така може во целост да се превземе од интернет.

3.7 Вчитување на звук во Python

Вчитувањето на аудио фајлови во Python се прави со употреба на модулот SciPy. Да го вчитаме нашиот прв звучен фајл:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.io import wavfile
fs, skopsko = wavfile.read('Skopsko_stereo.wav')
```

Со овој код во матрицата *skopsko* сме ги вчитале одбираците на дигитализираниот звук сместени во звучниот фајл „Skopsko_stereo.wav“. Променливата во која ги вчитуваме звуците во Python, во зависност од бројот на канали, може да има една или две вектор колони. Во нашиот случај станува збор за стерео датотека па соодветно променливата *skopsko* ќе има две вектор колони. Во *fs* ќе биде запишана фреквенцијата на одбирање со која е запишан звукот, а бројот на битови по примерок, односно должината на дигиталниот збор, може да се види од типот на променливите во матрицата *skopsko*.

За да го видиме ова променливи во работниот простор може да напишеме:

```
whos
```

Variable	Type	Data/Info
fs	int	44100
np	module	<module 'numpy' from '/usr<...>ages/numpy/_init__.pyc'>
plt	module	<module 'matplotlib.pyplot<...>s/matplotlib/pyplot.pyc'>
skopsko	ndarray	811782x2: 1623564 elems, type `int16`, 3247128 bytes (3 Mb)
wavfile	module	<module 'scipy.io.wavfile<...>es/scipy/io/wavfile.pyc'>

Гледаме дека *skopsko* има точно две вектор колони со по 811.782 примероци, што при *fs* од 44.100 kHz е аудио со должина од 18,4 s. Ова може да го пресметаме на следниот начин:

```
skopsko.shape[0]/fs
```

Out: 18

Овој резултат е добиен со целобройно делење. За да го добиеме точното времетраење можеме или *fs* да го кастуваме со **float**(*fs*) или да ја повикаме операцијата делење од Python 3.x:

```
from __future__ import division
skopsko.shape[0]/fs
```

²⁹The Python Tutorial. <https://docs.python.org/2.7/tutorial/index.html>

Out: 18.407755102040817

Освен димензиите на *skopsko* со `whos` може да видиме и колкав мемориски простор таа зафаќа – *skopsko* зафаќа околу 3 MB, колку што е и голем wav фајлот на дискот. Резолуцијата на кватнизијација е 16 битови, па затоа елементите на NumPy матрицата се од типот `int16`.

3.8 Скратување на звукот и префрлање во моно

Аудио материјалот сместен вака во матрицата *skopsko* може да се обработува како и сите други вектори и матрици во Python. За да го скратиме аудиосигналот на 4 s треба да внесеме:

```
skopsko = skopsko[:4*fs, :]
```

Постојат неколку начини да го направиме стерео аудиосигналот во моно. Наједноставно тоа може да се направи со задржување на само еден од каналите:

```
skopsko = skopsko[:, 0]
```

или:

```
skopsko = skopsko[:, 1]
```

Најправилниот начин тоа да се направи е преку усреднување на двата канали:

```
skopsko_mono = skopsko[:, 0] + skopsko[:, 1]
skopsko_mono /= 2
```

што може да се пресмета и како:

```
skopsko_mono = np.sum(skopsko, 1)
skopsko_mono /= 2
```

односно наједноставно со:

```
skopsko_mono = np.mean(skopsko, 1)
```

Во функциите `np.sum()` и `np.mean()`, вториот параметар кажува по која димензија да се изврши пресметката, со 1 е избрана првата димензија – односно по колони (хоризонтала). Со 0 може да се избере нултата димензија – односно по редици (вертикална), која е и зададена по default.

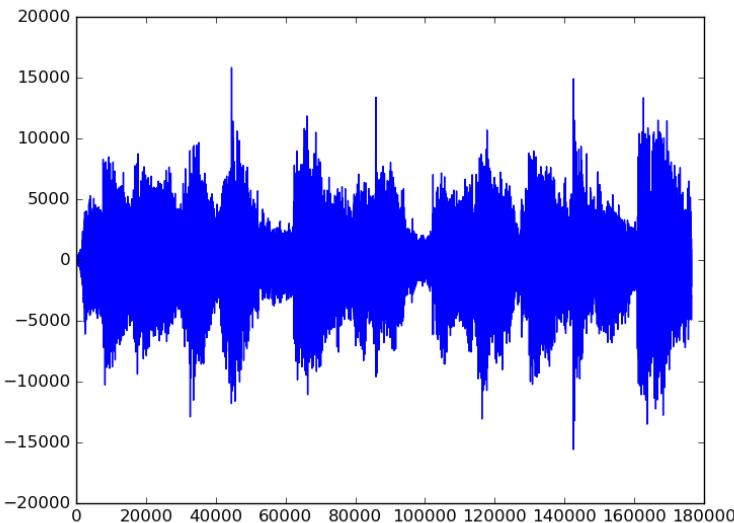
3.9 Прикажување на аудиосигналот

За да го прикажеме аудиосигналот ќе го искористиме `matplotlib`:

```
plt.plot(skopsko_mono)
plt.show()
```

Со што ќе биде исцртан графикот прикажан на Сл. 3.1. На овој график на *x*-оската е даден бројот на примерокот, а на *y*-оската неговата амплитуда. Бидејќи аудиото е со 16-битна резолуција амплитудите на примероците се движат соодветно од -2^{15} до $2^{15} - 1$.

За да ги доведеме примероците во опсег од -1 до 1, сè што треба да направиме е да го поделиме векторот *skopsko_mono* со 2^{15} :



Сл. 3.1: Приказ на аудиосигналот `skopsko_mono` како вектор од елементи од типот `int16`.

```
skopsko_mono /= 2**15
whos
```

Variable	Type	Data/Info
skopsko_mono	ndarray	176400: 176400 elems, type `float64`, 1411200 bytes (1 Mb)

Може да забележиме дека сега елементите од векторот не се веќе од типот `int16`, ами од типот `float64`. Сега да креираме временски вектор и да го прикажеме сигналот во време. Овој пат ќе ја искористиме опцијата `%matplotlib` за интерактивно плотирање.

```
skopsko_mono /= 2**15
t = np.arange(0,skopsko_mono.size) / fs
%matplotlib
plt.figure()
plt.plot(t, skopsko)
```

Со што ќе биде испртан графикот прикажан на Сл. 3.2.

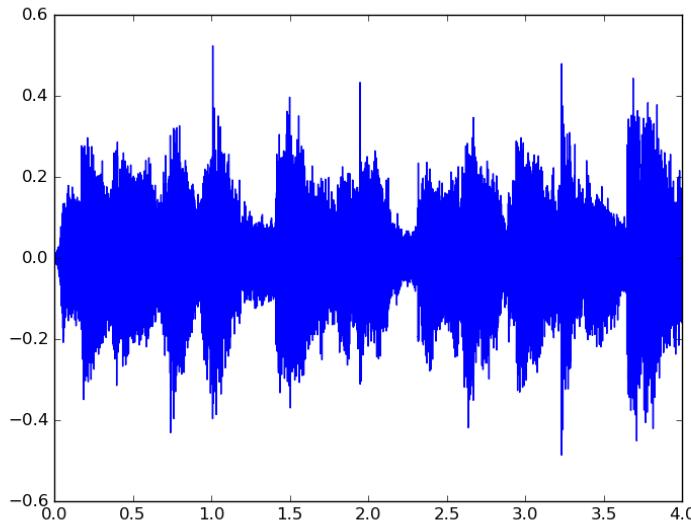
3.10 Преслушување на аудиосигналот

Наједноставниот начин да го преслушаме аудиосигналот е првин да го запишеме како wav фајл, а потоа да ја искористиме системската `play` наредба од SoX.

```
wavfile.write('Skopsko_mono.wav',fs, skopsko_mono)
!play Skopsko_mono.wav
```

`Skopsko_mono.wav:`

```
File Size: 1.41M      Bit Rate: 2.82M
Encoding: F.P. PCM
```



Сл. 3.2: Приказ на аудиосигналот *skopsko_mono* како вектор од елементи од типот `float64`.

```
Channels: 1 @ 53-bit
Samplerate: 44100Hz
Replaygain: off
Duration: 00:00:04.00
```

```
In:37.2% 00:00:01.49 [00:00:02.51] Out:65.5k [ ==|====== ] Clip:0
```

Може да забележиме дека Python го запишал аудио фајлот со 64-битна прецизност наместо во оригиналната 16-битна. За ова да го поправиме треба да напишеме:

```
wavfile.write('Skopsko_mono.wav',fs,np.array(skopsko_mono * 2**15, dtype='int16'))
!play Skopsko_mono.wav
```

Skopsko_mono.wav:

```
File Size: 353k      Bit Rate: 706k
Encoding: Signed PCM
Channels: 1 @ 16-bit
Samplerate: 44100Hz
Replaygain: off
Duration: 00:00:04.00
```

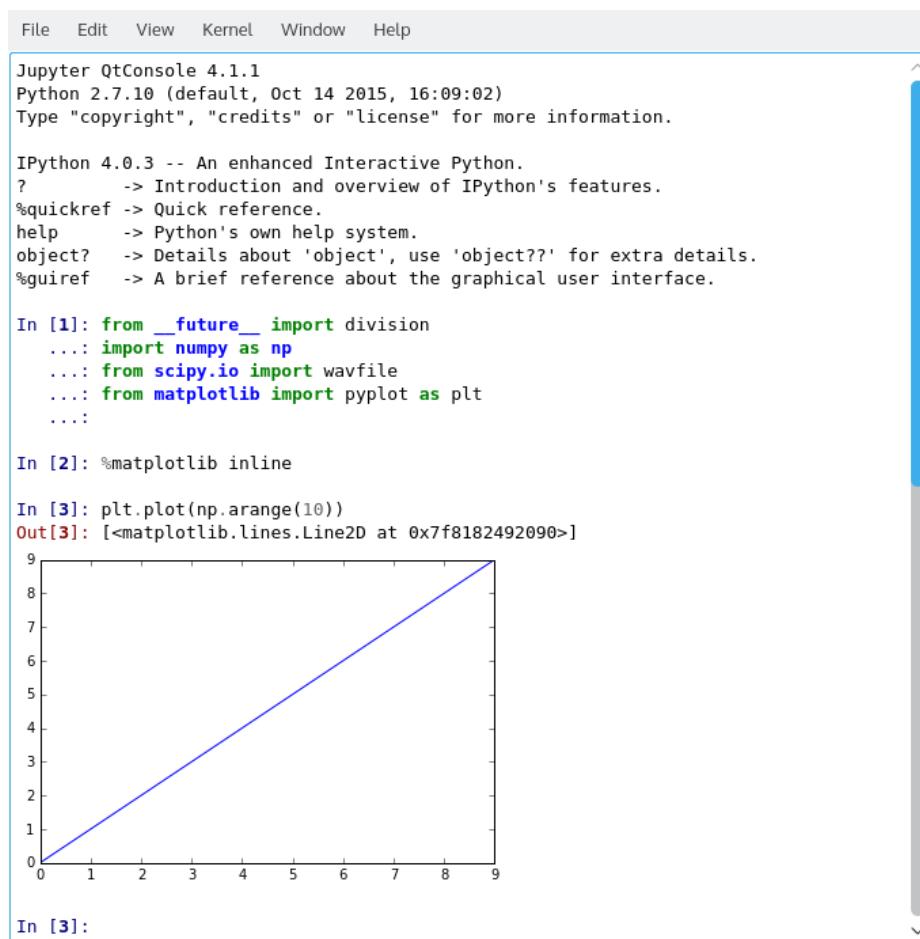
```
In:100% 00:00:04.00 [00:00:00.00] Out:176k [ -==|===== ] Clip:0
Done.
```

3.11 Менување на амплитудата на аудиосигналот

За овој дел од вежбата ќе ја употребиме моќната **Jupyter QtConsole** прикажана на Сл. 3.3 реализирана во Qt технологијата. Таа нуди боене на клучните зборови од кодот, автоматска помош при работа со функции, како и плотирање во самата конзола. За да ја стартувате во терминал напишете:

```
$ ipython qtconsole
```

Во неа треба повторно да ги вчитаме потребните пакети:



Сл. 3.3: IPython Qt конзолата нуди напредна интерактивност.

```

from __future__ import division
import numpy as np
from scipy.io import wavfile
from matplotlib import pyplot as plt
%matplotlib

```

§ Дополнително. Во Jupyter има опција и за плотирање во самата конзола која се избира со наредбата `%matplotlib inline`.

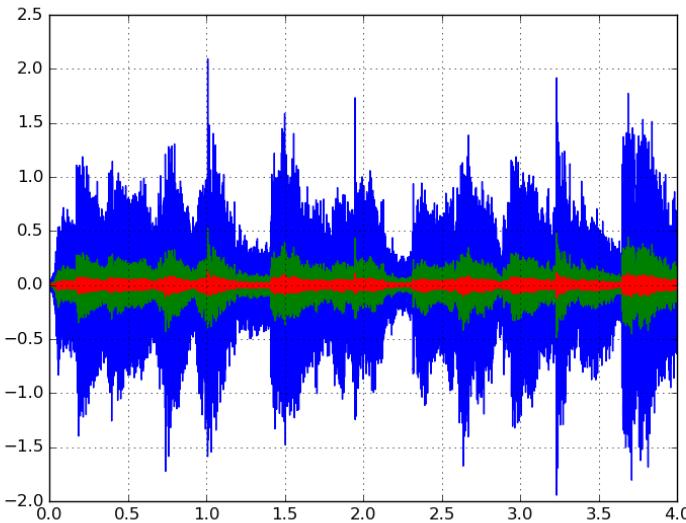
За да го засилиме или втишаме аудиосигналот сè што треба да направиме е да го помножиме со некој коефициент за скалирање.

```

skopsko_glasno = skopsko_mono * 4
skopsko_tivko = skopsko_mono * .25
plt.figure()
plt.plot(t, skopsko_glasno)
plt.plot(t, skopsko_mono)
plt.plot(t, skopsko_tivko)
plt.grid()

```

Добиениот график е даден на Сл. 3.4. Може да видиме дека засилениот сигнал `skopsko_glasno` има амплитуда над дозволениот опсег од -1 до 1 . Ова значи дека при снимањето на сигналот во wav фајл вредностите надвор од опсегот нема да бидат запишани, туку ќе бидат пресечени. Ова може да го чуеме со `play` наредбата.



Сл. 3.4: Приказ на аудиосигналот *skopsko_tono* скалиран со различни коефициенти.

```
wavfile.write('skopsko_glasno.wav', fs, np.array(skopsko_glasno, dtype='int16'))
!play skopsko_glasno.wav
```

Оваа појава на пресекување на амплитудите на звучниот сигнал се нарекува **дисторзија** и во некои типови на музика се користи како пожелен аудиоэффект. За да ја прикажеме дисторзијата графички може да напишеме:

```
skopsko_dist = skopsko_glasno.copy()
skopsko_dist[skopsko_dist > 1] = 1
skopsko_dist[skopsko_dist < -1] = -1
plt.figure()
plt.plot(t, skopsko_glasno)
plt.plot(t, skopsko_dist)
plt.grid()
```

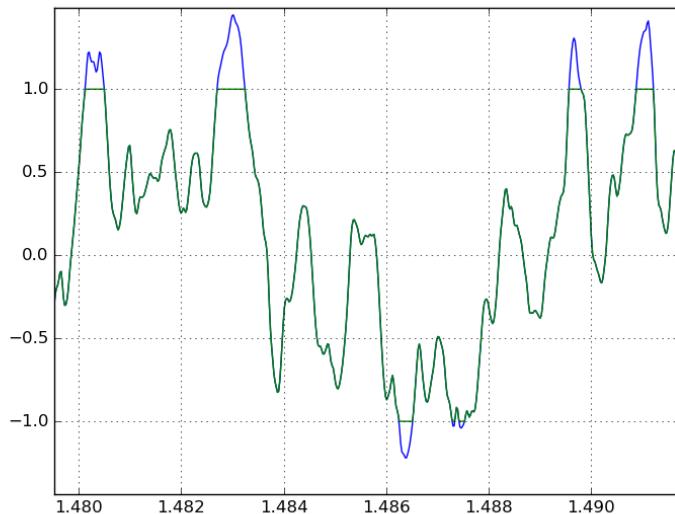
Копирањето во првата линија е нужно за да го задржиме оригиналниот вектор *skopsko_glasno*. Со зумирање може да го добиеме приказот на Сл. 3.5.

3.12 Нормализација на аудиосигналот

Во дигиталното процесирање на звукот, поради ограничениот опсег во којшто може да биде звучниот сигнал, од посебно значење е процесот на **нормализација**. Под нормализација се подразбира доведување на максималната амплитуда на аудиосигналот A_{max} на ниво зададено со:

$$L = 20 \log \frac{A_{max}}{1} [\text{dBFS}], \quad (3.1)$$

каде како референтно е земено максималното ниво во дигитален домен 1. Ова дигитално ниво на звукот се изразува во dBFS, што е кратенка од dB Full Scale, или полна скала. Од (3.1) следува дека за максималната дозволена амплитуда аудиосигналот има ниво од 0 dBFS. Во



Сл. 3.5: Приказ на дисторзија во аудиосигналот *skopsko_glasno*.

праксата вообичаено звучните сигнали се нормализираат на амплитуда помала од 1, за при нивното понатамошно процесирање или пренос да не дојде до дисторзија. Оваа амплитудна маргина што се остава до дозволениот максимум се нарекува **headroom**.

Да напишеме функција за нормализација:

```
def normalize(wav_in, level):
    amp_new = 10***(level/20)
    amp_max = max(abs(wav_in))
    return amp_new * wav_in / amp_max
```

која може да ја употребиме за нормализација на звучните сигнали:

```
skopsko_mono_0dBFS = normalize(skopsko_mono, 0)
skopsko_mono_3dBFS = normalize(skopsko_mono, -3)
skopsko_mono.max()
skopsko_mono_0dBFS.max()
skopsko_mono_3dBFS.max()
```

3.13 Генерирање на звук во Python

За да генерираме еден простопериодичен синусен тон на фреквенција од 200 Hz ќе напишеме:

```
from math import pi
sound = np.sin(2*pi*200*t)
plt.plot(sound)
wavfile.write('sin.wav', fs, np.array(sound * 2**15, dtype='int16'))
!play sin.wav
```

Овој код ќе го поместиме во скрипта `make_sound.py` која ќе може да ја извршуваме и директно од терминал. Отворете ново јазиче во терминалот со наредбата `Ctrl+Shift+T` и стартувајте го едиторот **Sublime Text** со наредбата:

```
$ subl make_sound.py &
```

Во скриптата внесете ги следните наредби:

```
from __future__ import division
import numpy as np
from scipy.io import wavfile
import sys # za vlezni argumenti
import os # za sistemski naredbi
from math import pi

print 'sys.argv : ', sys.argv # prikazi vlezni parametri

f = int(sys.argv[1])

fs = 44100
t = np.arange(0, 4*fs) / fs
sound = np.sin(2*pi*f*t)
wavfile.write('sin.wav',fs, np.array(sound * 2**15, dtype='int16'))
os.system('play sin.wav')
```

и повикајте ја од IPython³⁰:

```
%run make_sound.py 500
```

§Користејќи ја скриптата `make_sound.py` тестирајте ги границите на вашиот слух!

³⁰За да го смените јазичето искористете ја комбинацијата `Ctrl+PgUp/PgDwn`.

Поглавје 4

Фреквенциски спектар на звучните сигнали

Фреквенцискиот спектар на звучните сигнали е клучен во нивното поимање од страна на човекот. Спектарот на звучните сигнали е клучен и при нивната анализа, синтеза и обработка во дигиталните аудио системи. Со негова помош можеме да ги видиме карактеристиките на звучниот сигнал кои ги слушаме, а кои не се јасно видливи во неговиот временски облик. За пресметка на спектарот на аудиосигналите ќе се послужиме со **Фуриевата трансформација** и нејзината временски определена форма **STFT**¹. Ќе видиме како STFT анализата на звучниот сигнал води до една прегледна форма за претставување на неговите временско/спектрални особини – **спектрограмот**.

4.1 Основи на Фуриевата анализа

За потребите на овој курс, во ова поглавје ќе биде направен краток преглед на основите на Фуриевата анализа. Притоа ќе се задржиме само до особините и деталите кои се релевантни за дигиталното процесирање на аудиосигналите. За подетален преглед погледнете ја книгата Богданова (1997), по која се работи на предметот **Основи на дигитално процесирање на сигнали** на додипломските на **ФЕИТ**.

4.1.1 Фуриев интеграл

Фуриевата трансформација (FT) на сигналот $f(t)$ е дефинирана со интегралот:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt, \quad \omega \in (-\infty, \infty), \quad (4.1)$$

каде со ω е означена кружната фреквенција.

Инверзната Фуриева трансформација (IFT) е дефинирана со интегралот:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)e^{j\omega t} d\omega, \quad t \in (-\infty, \infty). \quad (4.2)$$

¹Short-Time Fourier Transform.

Табела 4.1: Позначајни својства на Фуреовата трансформација.

Својство	Временски домен	Фуреов домен
Линеарност	$af(t) + bg(t)$	$aF(\omega) + bG(\omega)$
Поместување	$f(t - t_0)$	$e^{-j t_0 \omega} F(\omega)$
Модулација	$e^{-j \omega_0 t} f(t)$	$F(\omega - \omega_0)$
Конволуција	$f(t) * g(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$	$F(\omega)G(\omega)$
Мултипликација	$f(n)g(n)$	$\frac{1}{2\pi} F(\omega) * G(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\Omega)G(\omega - \Omega)d\Omega$

$F(\omega)$ се нарекува **Фреквенциски спектар** на сигналот $f(t)$ и претставува комплексна функција од ω која може да се запише како:

$$F(\omega) = |F(\omega)|e^{j\angle F(\omega)} = A(\omega)e^{j\phi(\omega)}, \quad (4.3)$$

каде $A(\omega)$ претставува **амплитуден спектар**, а $\phi(\omega)$ **фазен спектар** на сигналот $f(t)$. Притоа, ако сигналот е реална функција од t тогаш амплитудниот спектар е парна функција од ω , а фазниот е непарна функција од t . Бидејќи интеграл од непарна функција во интервал симетричен околу координатниот почеток е 0, тогаш од (4.2), користејќи ја **Ојлеровата формула**, добиваме:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} A(\omega) e^{j\phi(\omega)} e^{j\omega t} d\omega, \quad (4.4)$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} A(\omega) [\cos(\omega t + \phi(\omega)) + j \sin(\omega t + \phi(\omega))] d\omega, \quad (4.5)$$

$$= \frac{1}{\pi} \int_0^{\infty} A(\omega) \cos(\omega t + \phi(\omega)) d\omega. \quad (4.6)$$

Основните својства на Фуреовата трансформација се дадени во Табелата 4.1.

Сигналот $f(t)$ и неговата Фуреова трансформација $F(\omega)$ чинат **Фуриев пар**. Во Табелата 4.2 се дадени некои позначајни Фуреови парови.

4.1.2 Фуриев ред

Ако функцијата $f(t)$ е периодична, односно ако важи:

$$f(t) \equiv f(t + T), \quad \forall t, \quad (4.7)$$

тогаш ако со $\hat{f}(t)$ го означиме сегментот на $f(t)$ во основниот интервал $t \in [-T/2, T/2]$, т.е.:

$$\hat{f}(t) = f(t) \cdot rect(t), \quad (4.8)$$

каде

Табела 4.2: Позначајни Фуриеови парови.

$f(t)$	$F(\omega)$
$\delta(t)$	1
1	$2\pi\delta(\omega)$
$rect(t) = \begin{cases} 1 & t < T \\ 0 & t > T \end{cases}$	$2T \frac{\sin(T\omega)}{T\omega}$
$\frac{\sin(\Omega t)}{\pi t}$	$rect(\omega) = \begin{cases} 1 & \omega < \Omega \\ 0 & \omega > \Omega \end{cases}$
$\cos(\omega_0 t)$	$\pi [\delta(\omega + \omega_0) + \delta(\omega - \omega_0)]$
$\sin(\omega_0 t)$	$j\pi [\delta(\omega + \omega_0) - \delta(\omega - \omega_0)]$
$rect(t) = \begin{cases} 1, & t < T, \\ 0, & t > T. \end{cases}$	

Можеме да ја изразиме $f(t)$ како сума од вакви функции поместени за мултипили од T :

$$f(t) = \sum_{r=-\infty}^{\infty} \hat{f}(t - rT). \quad (4.10)$$

Во тој случај во Фуриев домен, функцијата наместо да биде контуирана по кружната фреквенција ω , таа ќе биде еднаква на сума од комплексни синусоиди која се нарекува и **Фуриев ред**:

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\frac{2\pi}{T}t} = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}, \quad (4.11)$$

каде:

$$\omega_0 = \frac{1}{T}. \quad (4.12)$$

Фреквенциите на комплексните синусоиди претставуваат мултипили од основната фреквенција на сигналот ω_0 . Притоа коефициентите c_k кои претставуваат амплитуди на комплексните синусоиди можат да се добијат преку:

$$c_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-jk\frac{2\pi}{T}t}. \quad (4.13)$$

Фреквенцискиот спектар на ваков периодичен сигнал $f(t)$ претставуваsuma од поместени евидистантни Диракови импулси $\delta(\omega)$:

$$F(\omega) = 2\pi \sum_{k=-\infty}^{\infty} c_k \delta(\omega - k\omega_0). \quad (4.14)$$

Табела 4.3: Позначајни својства на Фуреовата трансформација.

Својство	Временски домен	Фуреов домен
Линеарност	$ax_1(t) + bx_2(t)$	$aX_1(z) + bX_2(z)$
Поместување	$x(n - n_0)$	$z^{n_0}X(z)$
Конволуција	$x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$	$X(z)H(z)$
Мултипликација	$x(n)w(n)$	$\frac{1}{2\pi j} \oint_C X(\nu)W(z/\nu)\nu^{-1}d\nu$

Табела 4.4: Позначајни Z парови.

$x(n)$	$X(z)$
$\delta(n - n_0)$	z^{-n_0}
$rect(n) = \begin{cases} 1, & 0 \leq n < N, \\ 0, & \text{поинаку.} \end{cases}$	$\sum_{n=0}^{N-1} z^{-n} = \frac{1 - z^{-N}}{1 - z^{-1}}$
$a^n u(n)$	$\frac{1}{1 - z^{-1}}, \quad a < z $

4.1.3 Z-трансформација

За да видиме како Фуреовата анализа може да се примени врз дискретни сигнали ќе ја воведеме **Z-трансформацијата** (Rabiner and Schafer, 1978) дефинирана со:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}, \quad (4.15)$$

каде $x(n)$ е дискретна верзија од континуираната функција $f(t)$:

$$x(n) = f(nT_s), \quad (4.16)$$

$$T_s = \frac{1}{F_s}. \quad (4.17)$$

тука F_s е фреквенцијата на семплирање, а T_s периодот на семплирање.

Инверзната Z-трансформација е дефинирана со:

$$x(n) = \frac{1}{2\pi j} \oint_C X(z)z^{n-1}dz, \quad (4.18)$$

каде C е затворена контура во z -рамнината која го вклучува центарот, а се наоѓа во пределот на конвергенција на трансформацијата $z \in (R_1, R_2)$. Позначајни особини на Z-трансформацијата се дадени во Табелата 4.3, а позначајни Z-трансформации во Табелата 4.4.

4.1.4 Фуреова трансформација на дискретен сигнал

Фуреовата трансформација на дискретен сигнал може да се добие од (4.15) со замената $z = e^{-j\omega n}$:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}, \quad (4.19)$$

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega)e^{j\omega n} d\omega. \quad (4.20)$$

Во овој случај z е ограничен на единечната кружница во z -рамнината а дигиталната кружна фреквенција ω има интерпретација на аголот во оваа рамнина. Од тута може да се види една од важните особини на Фуриевата трансформација, а тоа е нејзината периодичност во однос на ω со период 2π .

* **Важно!** Фуриевата трансформација $X(\omega)$ на дискретниот сигнал $x(n)$ е сеуште континуирана функција од ω !

4.1.5 Дискретна Фуриева трансформација

Како што беше случај во аналоген домен, ако еден дискретен сигнал е периодичен со периода N , односно:

$$\tilde{x}(n) = \tilde{x}(n + N), \quad -\infty < n < \infty, \quad (4.21)$$

тогаш $\tilde{x}(n)$ во Фуриев домен ќе биде претставена како дискретна сума од синусоиди:

$$\tilde{X}(k) = \sum_{n=0}^{N-1} \tilde{x}(n)e^{-jk\frac{2\pi}{N}n}, \quad (4.22)$$

$$\tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k)e^{jk\frac{2\pi}{N}n}. \quad (4.23)$$

Ако сега ја земеме Фуриевата трансформација на дискретниот сигнал $x(n)$ дадена во (4.19), која претставува Z -трансформација на сигналот $x(n)$ долж единечната кружница, и истата ја семплираме на еквидистантни фреквенции ω_k дадени со:

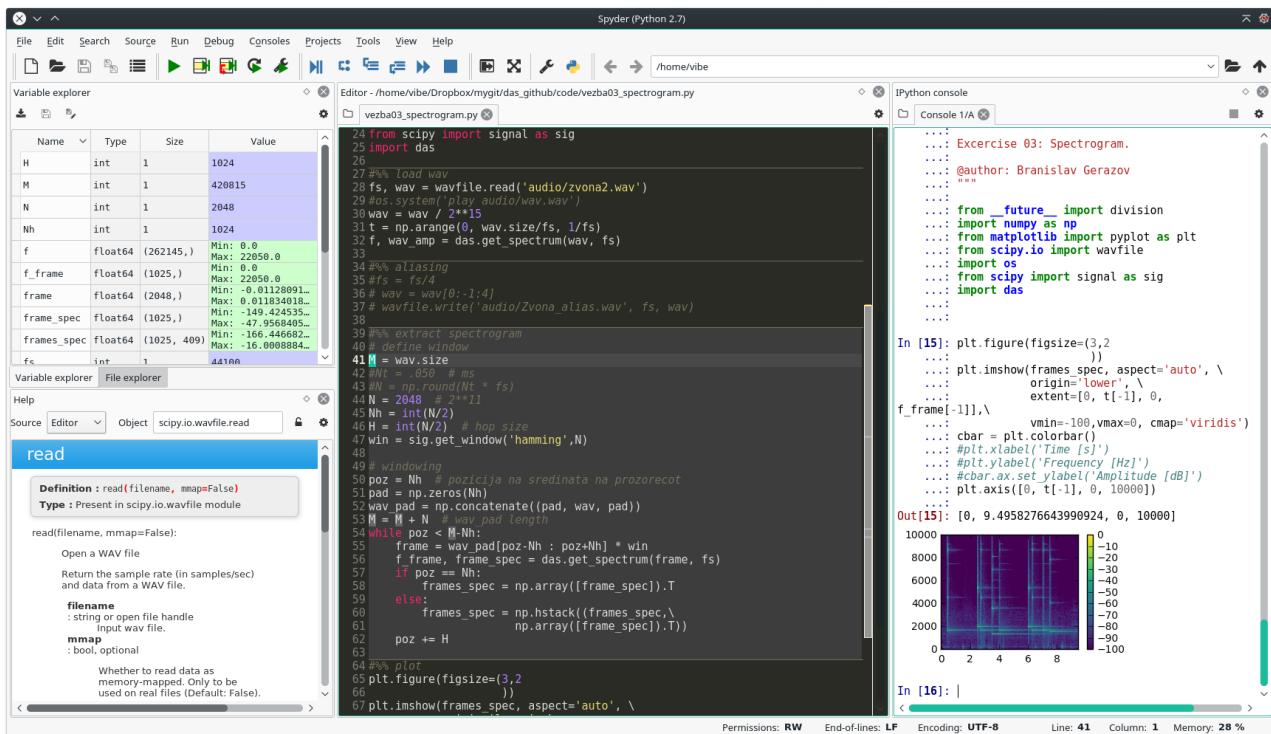
$$\omega_k = k \frac{2\pi}{N}, \quad (4.24)$$

тогаш ќе го добиеме изразот за **дискретната Фуриева трансформација DFT**:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-jk\frac{2\pi}{N}n}, \quad (4.25)$$

кој е еквивалентен со (4.22)! Ова значи дека семплиријаниот спектар што го добиваме со (4.25) соодветствува на сигнал $\tilde{x}(n)$ која претставува периодична верзија од сигналот $x(n)$ со период N , односно важи:

$$x(n) = \tilde{x}(n) \cdot \text{rect}(n), \quad (4.26)$$



Сл. 4.1: Развојната средина за Python **Spyder** специјализирана за научни истражувања.

каде

$$\text{rect}(n) = \begin{cases} 1, & 0 \leq n < N, \\ 0, & \text{поинаку.} \end{cases} \quad (4.27)$$

Соодветно **инверзната дискретна Фурьеова трансформација IDFT** е дадена со изразот:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{jk \frac{2\pi}{N} n}. \quad (4.28)$$

За пресметка на DFT во праксата се користат алгоритми за нејзино брзо пресметување чија пресметковна сложеност е пропорционална со $N \log N$, кои се нарекуваат **брза Фурьеова трансформација FFT**.

4.2 Анализа на спектарот на звучните сигнали

4.2.1 Spyder

За анализа на спектарот на звучните сигнали ќе ја воведеме развојната средина за Python специјализирана за научни истражувања **Spyder**² прикажана на Сл. 4.1.

Spyder во себе вклучува:

- **Едитор** – со вклучен браузер на функции/класи, можности за анализа на код, автоматско завршување на код, и вчитување на дефиниции.

²Spyder - The Scientific PYthon Development EnviRonment. <https://github.com/spyder-ide/spyder>

- **Интерактивна конзола** – интегрирани Python и IPython конзоли со работни простории и поддршка за дебагирање и поддршка за Matplotlib, овозможуваат инстантна евалуација на кодот напишан во едиторот.
- **Документација** – покажување на документацијата на било која класа или функција повикана во едиторот или конзолата.
- **Приказ на променливи** – овозможува брза анализа на променливите генериирани со некој код.
- **Приказ на фајлови и фолдери.**
- **Историја на наредби.**

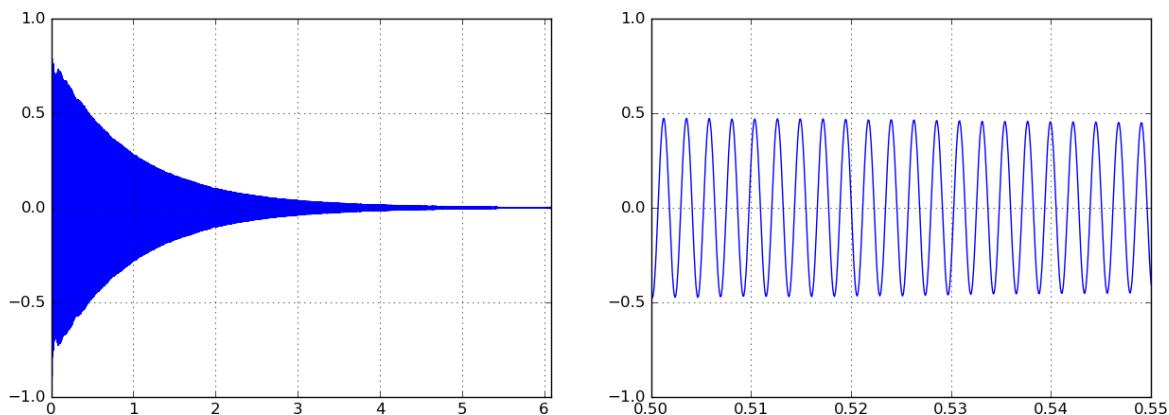
4.2.2 Спектар на простопериодичен звук

Да го пресметаме и прикажеме спектарот на еден простопериодичен, односно синусоиден, тон ќе го искористиме звукот генериран од звучната вилушка, којшто е еден од ретките природни звуци со простопериодична природа. Снимка од овој сигнал имаме во фајлот *Viluska 440Hz.wav* кој е дел од аудио записите од предметот **Електроакустика** кои ќе ги искористиме за илустрација на различните типови на спектри во оваа вежба.

За почеток, фајлот кој автоматски се отвора со стартирање на Spyder да го снимиме под името *vezba02_spekar.py* во фолдерот *das*. Во него да ги увеземе основните модули и да го вчитаме, преслушаме и прикажеме аудиосигналот.

```

1 # -*- coding: utf-8 -*-
2 """
3 Дигитални аудио системи
4 Вежба 2 - спектар
5 Created on Wed Mar 23 23:21:08 2016
6 @author: das
7 """
8
9 from __future__ import division
10 import numpy as np
11 import matplotlib.pyplot as plt
12 from scipy.io import wavfile
13 import os
14
15 filename = 'audio/Viluska_440Hz.wav'
16 fs, viluska = wavfile.read(filename)
17 os.system('play ' + filename)
18
19 viluska = viluska / 2**15
20 N = viluska.size
21 t = np.arange(0, N)
22 t = t / fs
23 plt.figure(figsize=(15,5))
24 plt.subplot(121) # 1X2 графици, прв график
25 plt.plot(t, viluska)
26 plt.grid()
27 plt.axis([0, t[-1], -1, 1]) # [xmin, xmax, ymin, ymax]
```



Сл. 4.2: Временски облик на аудиосигналот генериран од звучна вилушка.

```

27 plt.subplot(122) # 1x2 графици, втор график
28 plt.plot(t, viluska)
29 plt.grid()
30 plt.axis([0.5, 0.55, -1, 1]) # [xmin, xmax, ymin, ymax]

```

Графиците кои ќе ги генерира дадениот код се прикажани на Сл. 4.2

* **Важно!** При првото извршување на python скрипта во едиторот на Spyder треба да одберете каде да биде извршен кодот. Треба да ја одберете првата опција: Execute in current Python or IPython console. На овој начин сите променливи генериирани во кодот ќе бидат вчитани во сегашниот работен простор и ќе бидат на располагање по завршувањето на скриптата.

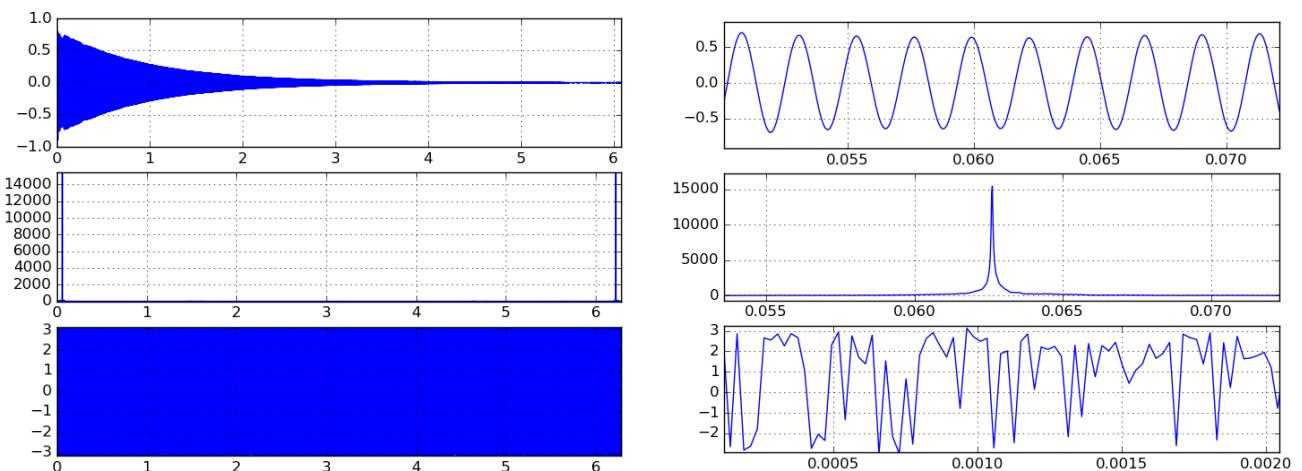
§ Дополнително. Графиците во Spyder можат да бидат исцртани во самата IPython конзола или интерактивно во посебен прозорец. За да се избере второво, треба да се направат потребните подесувања во Tools > Preferences > IPython console > Graphics > Backend : Automatic.

За да го пресметаме спектарот на овој звучен сигнал ќе ја искористиме имплементацијата на FFT алгоритамот во SciPy модулот **fftpack**, кој содржи и низа други функции за Фурьеова анализа. Во нашиот код ја додаваме секвенцата **#%%** што во Spyder претставува нова ќелија, а со тоа овој дел од скриптата ќе може едноставно да биде извршен без да се извршува целиот код со притискање **Ctrl+Enter**.

```

31 #%%
32 from scipy import fftpack as fp
33
34 viluska_fft = fp.fft(viluska)
35 viluska_amp = np.abs(viluska_fft)
36 viluska_ph = np.angle(viluska_fft)
37 w = np.arange(0, 2*pi, 2*pi/N)
38
39 plt.figure()
40 plt.subplot(311)
41 plt.plot(t, viluska)
42 plt.axis([0, t[-1], -1, 1]) # [xmin, xmax, ymin, ymax]
43 plt.grid()

```



Сл. 4.3: Амплитуден (средина) и фазен (долу) спектар на аудиосигналот генериран од звучна вилушка (ропе).

```

44 plt.subplot(312)
45 plt.plot(w, viluska_amp)
46 plt.axis([0, w[-1], np.min(viluska_amp), np.max(viluska_amp)])
47 plt.grid()
48 plt.subplot(313)
49 plt.plot(w, viluska_ph)
50 plt.axis([0, w[-1], np.min(viluska_ph), np.max(viluska_ph)])
51 plt.grid()

```

Графиците добиени со овој код се прикажани на Сл. 4.3. Може да забележиме дека амплитудниот и фазниот спектар се движат од 0 до 2π , а амплитудата на вториот оди од $-\pi$ до π . Вообичаено двета спектри се прикажуваат во однос на фреквенција во Hz; амплитудниот спектар вообичаено се изразува во dB, што е поблиску до начинот на кој човекот го поима звукот; фазниот спектар нужно се одмотува од опсегот $-\pi$ до π и покрај неговата периодичност за да се воочи подобро неговата динамика.

Уште повеќе, поради тоа што за реални сигнали, какви што се аудиосигналите, половина од спектарот е огледална слика на првата половина, таа може да ја занемариме. Конечно, за побрза работа на FFT должината на влезниот сигнал треба да биде степен од 2. За таа цел, сигналот вообичаено се дополнува со нули до следната поголема должина која е степен од 2, што може да се пресмета на следниот начин:

$$N_{FFT} = 2^x, \quad (4.29)$$

каде:

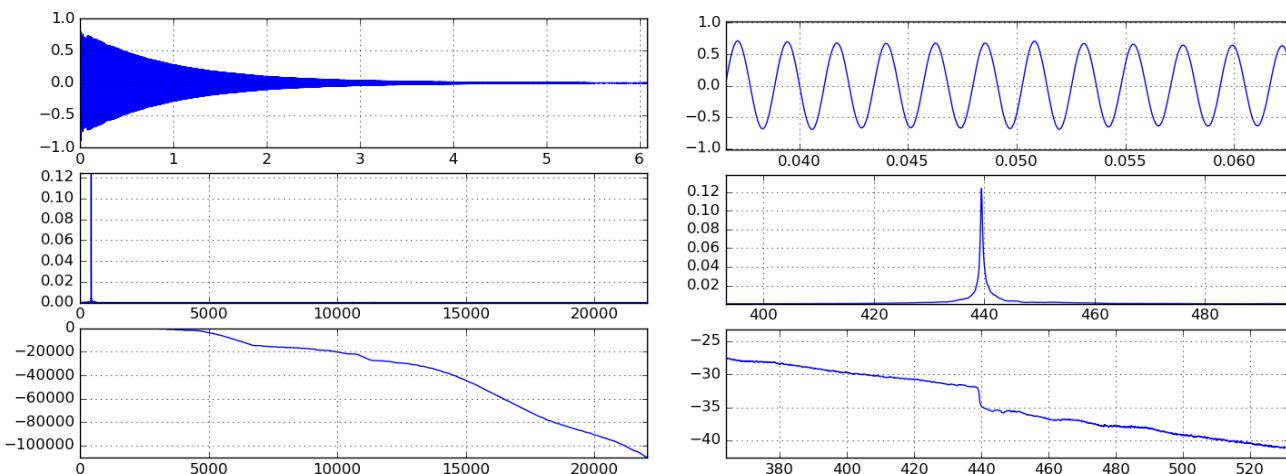
$$x = \lceil \log_2 N \rceil \quad (4.30)$$

а N е должината на сигналот. Сите овие работи може да ги направиме со следниот код:

```

31 #%%
32 from scipy import fftpack as fp
33 Nfft = 2**np.ceil(np.log(N)/np.log(2))
34 Nh = Nfft / 2
35 viluska_fft = fp.fft(viluska, Nfft)
36 viluska_fft = viluska_fft[0:Nh+1]
37 viluska_fft = viluska_fft / N
38 viluska_fft[2:-1] = viluska_fft[2:-1] * 2
39 viluska_amp = np.abs(viluska_fft)
40 viluska_amp = 20*np.log10(viluska_amp)

```



Сл. 4.4: Вообичаен приказ ма амплитудниот (средина) и фазниот (долу) спектар на аудиосигналот генериран од звучна вилушка (горе).

```

41 viluska_ph = np.angle(viluska_fft)
42 viluska_ph = np.unwrap(viluska_ph)
43 f = np.arange(0, Nh+1)
44 f = f / Nfft *fs
45 ...

```

Притоа сме направиле скалирање на FFT коефициентите со должината на сигналот N , а амплитудите на коефициентите кои сме ги отфрлиле сме ги додале на нивните огледални слики. Добиениот приказот на двата спектри на аудиосигналот е даден на Сл. 4.4.

✓ **Задача за час.** Проверете што работи функцијата `fftshift` од `scipy.fftpack`.

Бидејќи пресметката на спектарот на аудиосигналите е од суштествено значење во дигиталните аудиосистеми, можеме дадениот код да го искористиме за пишување на функција за таа намена. Функцијата ќе ја поместиме во нов модул кој ќе го наречеме `das.py` со следната содржина:

```

from __future__ import division
import numpy as np
from scipy import fftpack as fp

def get_spectrum(wav, fs):
    N = wav.size
    Nfft = 2**np.ceil(np.log(N)/np.log(2))
    Nfft = 4
    wav_spec = fp.fft(wav)
    wav_amp = np.abs(wav_spec)
    wav_amp = wav_amp / N
    wav_amp = wav_amp[0:Nfft/2+1]
    wav_amp[2:-1] = wav_amp[2:-1] * 2
    wav_amp = 20*np.log10(wav_amp)
    f = np.linspace(0, fs/2, Nfft/2+1)
    return f, wav_amp

```

За користење на новата функција `get_spectrum()` сè што треба да направиме е да го увеземе модулот исто како и другите Така `viluska_amp` може сега да го добиеме на следниот начин:

```

import das
f, viluska_amp = das.get_spectrum(viluska, fs)

```

4.3 Фуриеова трансформација на временски отсекоци STFT

Главниот недостаток на Фуриеовата трансформација е тоа што таа не ни дава никаква временска информација за сигналот кој што го анализираме. Поради нестационарната природа на аудиосигналите, Фуриеовата анализа на целиот сигнал може да ни каже некои општи карактеристики за сигналот, но не може да ни ги покаже деталите. На пример, вкупниот спектар на еден музички сигнал може да ни каже дали во него има изразен бас преку анализата на енергијата на сигналот во ниските фреквенции, но ваквиот вкупен спектар не може да ни каже колку е брз ритамот на музичкиот сигнал. Оттаму потребата за временска локализација на спектралната информација што ја нуди **Фуриеовата трансформација на временски отсекоци STFT**.



§ Дополнително. Унгарскиот електроинженер **Денес Габор**³ (1900–1979) е оној кој во 1946 ја адаптира Фуриеовата трансформација за временска анализа и ја добива STFT. Тој е и изумителот на холографијата за што добива Нобелова награда по Физика во 1971. Една од неговите најпознати изјави е: „Најдобриот начин да се предвиди иднината е таа да се создаде.“

STFT го анализира спектарот во низа од кратки временски отсекоци наречени **рамки**, земени од аудиосигналот со помош на функција за селекција $w(n)$ која се нарекува **прозорец**, според равенствето (4.31). Тука со i е означен редниот број на рамката, N е големината на прозорецот, а H е големината на скокот кој го прави прозорецот долж сигналот од една рамка до друга. Оваа метода на анализа на нестационарните сигнали преку земање на отсекоци со лизгање на прозорец по нивната должина се нарекува и **метода на прозорци**.

$$X_i(k) = \sum_{n=-N/2}^{N/2-1} w(n)x(n+i \cdot H)e^{-jk\frac{2\pi}{N}n}, \quad i = 0, 1, 2, \dots \quad (4.31)$$

На овој начин наместо еден вкупен спектар, се добива низа од спектри од сигналот пресметани за различни временски моменти во избрана нивна околина.

За да видиме што се случува при STFT ќе направиме анализа на еден простопериодичен сигнал $x(n)$:

$$x(n) = A \cos(\omega_0 n) = A \cos(2\pi f_0 n) = A \cos(2\pi \frac{k_0}{N} n) = \frac{A}{2} e^{j2\pi \frac{k_0}{N} n} + \frac{A}{2} e^{-j2\pi \frac{k_0}{N} n}, \quad (4.32)$$

ако замениме во (4.31) добиваме:

$$X(k) = \sum_{n=-N/2}^{N/2-1} w(n) \left(\frac{A}{2} e^{j2\pi \frac{k_0}{N} n} + \frac{A}{2} e^{-j2\pi \frac{k_0}{N} n} \right) e^{-jk\frac{2\pi}{N}n}, \quad (4.33)$$

$$= \frac{A}{2} \sum_{n=-N/2}^{N/2-1} w(n) e^{j2\pi \frac{k-k_0}{N} n} + \frac{A}{2} \sum_{n=-N/2}^{N/2-1} w(n) e^{-j2\pi \frac{k+k_0}{N} n}, \quad (4.34)$$

$$= \frac{A}{2} W[k - k_0] + \frac{A}{2} W[k + k_0], \quad (4.35)$$

каде $W(k)$ е Фуриеовата трансформација на прозорецот која е поместена фреквенциски за фреквенцијата на синусната функција. Оваа го потврдува својството на Фуриеовата трансформација според кое множењето на два сигнали во временски домен претставува нивна конволуција во Фуриеов домен.

³Wikipedia: Dennis Gabor. https://en.wikipedia.org/wiki/Dennis_Gabor

4.3.1 Видови на прозорци

Претходната анализа исто така го илустрира фактот дека спектарот на прозорецот има критично влијание за точната претстава на спектарот на аудиосигналите. Поради оваа причина дизајнирани се низа од различни видови на прозорци, секој со различни спектрални карактеристики, а со тоа и со различна примена.⁴ Сите тие во временски домен можат да се претстават како сума од косинуси, а во спектрален домен како сума од $\text{sinc}(n)$ функции.

Во STFT анализата на аудиосигналите најупотребувани прозорци (Serra and O Smith III, 2014) се:

- Правоаголен прозорец

$$\text{rect}(n) = 1, \quad -N/2 \leq n \leq N/2 \quad (4.36)$$

$$\text{Rect}(k) = \text{sinc}(k) \quad (4.37)$$

- Хан прозорец

$$\text{hann}(n) = 0,5 + 0,5 \cos\left(2\pi \frac{n}{N}\right), \quad -N/2 \leq n \leq N/2 \quad (4.38)$$

$$\text{Hann}(k) = 0,5 \text{sinc}(k) + 0,25 (\text{sinc}(k-1) + \text{sinc}(k+1)) \quad (4.39)$$

- Хаминг прозорец

$$\text{hamm}(n) = 0,54 + 0,46 \cos\left(2\pi \frac{n}{N}\right), \quad -N/2 \leq n \leq N/2 \quad (4.40)$$

- Блекман прозорец

$$\text{black}(n) = 0,42 - 0,5 \cos\left(2\pi \frac{n}{N}\right) + 0,080,5 \cos\left(4\pi \frac{n}{N}\right), \quad -N/2 \leq n \leq N/2 \quad (4.41)$$

- Блекман-Харис прозорец

$$\text{blackharr}(n) = \frac{1}{N} \sum_{i=0}^3 a_i \cos\left(2i\pi \frac{n}{N}\right), \quad -N/2 \leq n \leq N/2 \quad (4.42)$$

каде

$$a_0 = 0,35876, a_1 = 0,48829, a_2 = 0,14128, a_3 = 0,01168 \quad (4.43)$$

4.3.2 Карактеристики на прозорците

Најважните спектрални карактеристики на прозорците се **ширината на главното крило** и **амплитудата на најголемото споредно крило**. Идеалниот прозорец има бескрајно тесно главно крило и нема споредни крила. Во реалноста станува збор за компромис помеѓу овие два параметри. Различните видови на прозорци можеме да ги генерираме со функцијата `get_window()` од модулот `scipy.signal`. Во следниот код тоа е направено за правоаголниот прозорец `boxcar`. Притоа, амплитудните спектри се нормализираат до 0 dB.

⁴Wikipedia: Windowing function https://en.wikipedia.org/wiki/Window_function

```

1 # -*- coding: utf-8 -*-
2 from __future__ import division
3 import numpy as np
4 from matplotlib import pyplot as plt
5 from scipy import fftpack as fp
6 from scipy import signal as sig
7
8 M = 512 # околина за анализа
9 N = 64 # должина на прозорец
10 Mh = M/2
11 Nh = N/2
12 w = np.zeros(M)
13 w[Mh-Nh:Mh+Nh] = sig.get_window('boxcar', N)
14 W = fp.fft(w, M)
15 Wamp = np.abs(W)/N
16 eps = np.finfo(float).eps
17 Wamp[np.where(Wamp < eps)] = eps
18 Wlog = 20*np.log10(Wamp)
19 Wlog -= np.max(Wlog)
20 Wshift = fp.fftshift(Wlog)
21
22 plt.figure(figsize=(12,5))
23 plt.subplot(121)
24 plt.plot(w)
25 plt.axis([0, M, 0,1]) #[xmin, xmax, ymin, ymax]
26 plt.grid()
27 plt.subplot(122)
28 plt.plot(Wshift)
29 plt.grid()
30 plt.axis([0, M, -100,0]) #[xmin, xmax, ymin, ymax]

```

Различните прозорци и нивните амплитудни спектри генериирани со овој код се прикажани на Сл. 4.5. Може да се види дека правоагочниот прозорец иако е добар за процесирање во временски домен има најлоши спектрални карактеристики во поглед на амплитудата на страничните крила. Од друга страна, најмало влијание на страничните крила имаме кај Блакман-Харис прозорецот, но тој за сметка на тоа има најшироко главно крило. Хановиот и Хаминговиот прозорец имаат иста широчина на главното крило на различно распоредена енергија во страничните крила.

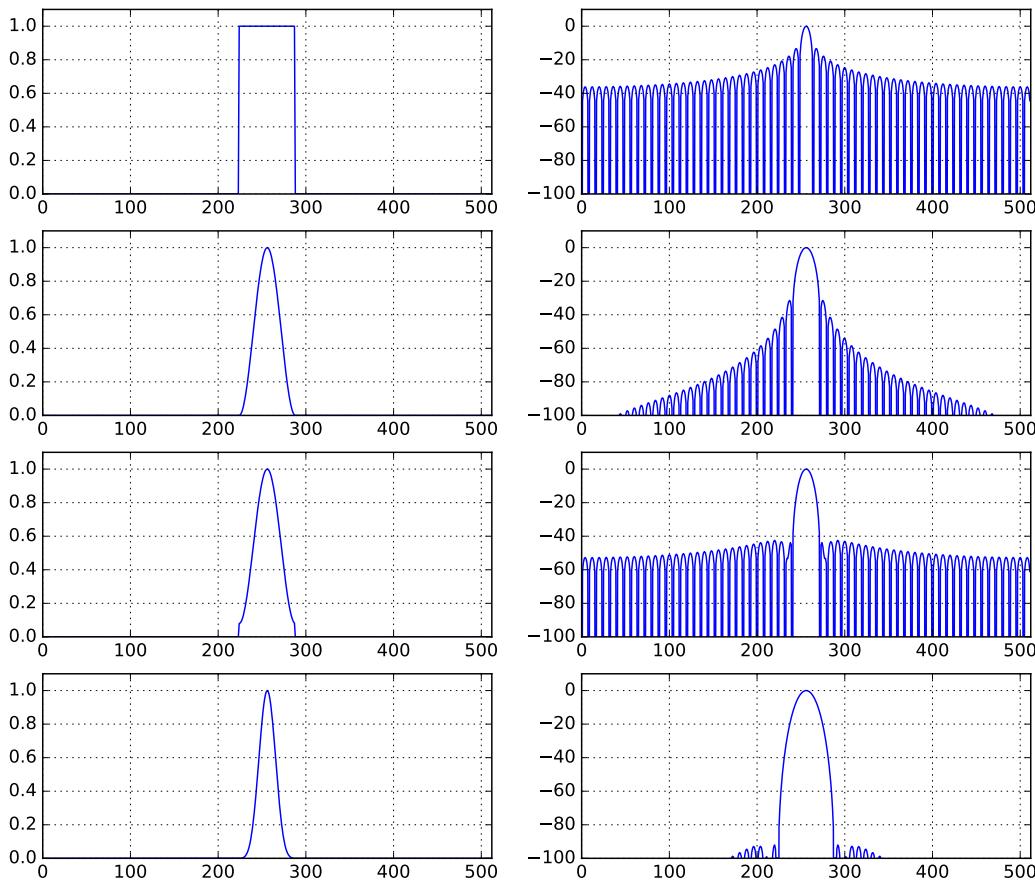
4.3.3 Спектrogram

Спектrogramот е еден од најзначајните прикази на аудиосигналите. На него во исто време може да се набљудуваат карактеристиките на сигналот и во временски и во спектрален домен. За да го собиеме ќе се послужиме со STFT. Приказот е даден на Сл. 4.6.

```

1 import ...
2 import das
3
4 fs, wav = wavfile.read('audio/Zvona.wav')
5 wav = wav / 2**15
6 t = np.arange(0, wav.size/fs, 1/fs)
7 N = 2048 # 2**11
8 Nh = int(N/2)
9 H = int(N/2) # hop size

```

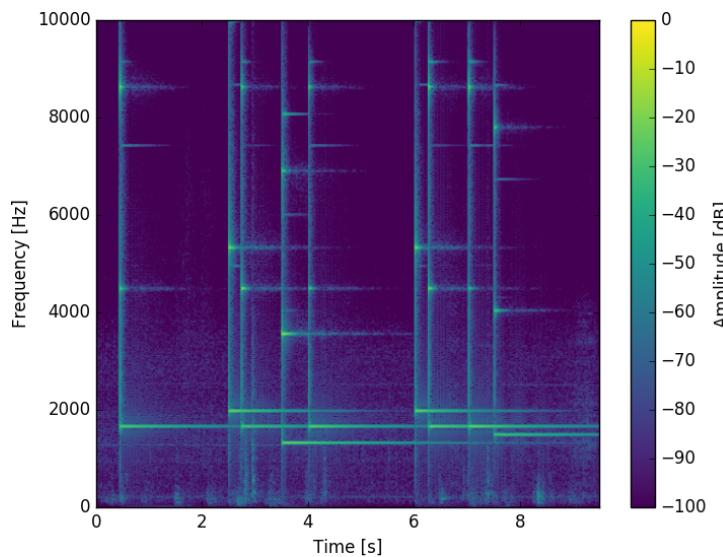


Сл. 4.5: Споредба на временскиот и спектрален облик на четири најчесто користени прозорци (од горе надолу): правоаголен, Ханов, Хамингов и Блекман-Харисов.

```

10 win = sig.get_window('hamming', N)
11 poz = Nh # pozicija na sredinata na prozorecot
12 pad = np.zeros(Nh)
13 wav_pad = np.concatenate((pad, wav, pad))
14 M = wav_pad.size
15 while poz < M-Nh:
16     frame = wav_pad[poz-Nh : poz+Nh] * win
17     f_frame, frame_spec = das.get_spectrum(frame, fs)
18     if poz == Nh:
19         frames_spec = np.array([frame_spec]).T
20     else:
21         frames_spec = np.hstack((frames_spec,\n22                         np.array([frame_spec]).T))
23     poz += H
24
25 plt.figure()
26 plt.imshow(frames_spec, aspect='auto', \
27             origin='lower', \
28             extent=[0, t[-1], 0, f_frame[-1]], \
29             vmin=-100, vmax=0, cmap='viridis')
30 cbar = plt.colorbar()
31 plt.xlabel('time [s]')
32 plt.ylabel('frequency [Hz]')
33 cbar.ax.set_ylabel('Amplitude [dB]')
34 plt.axis([0, t[-1], 0, 10000])

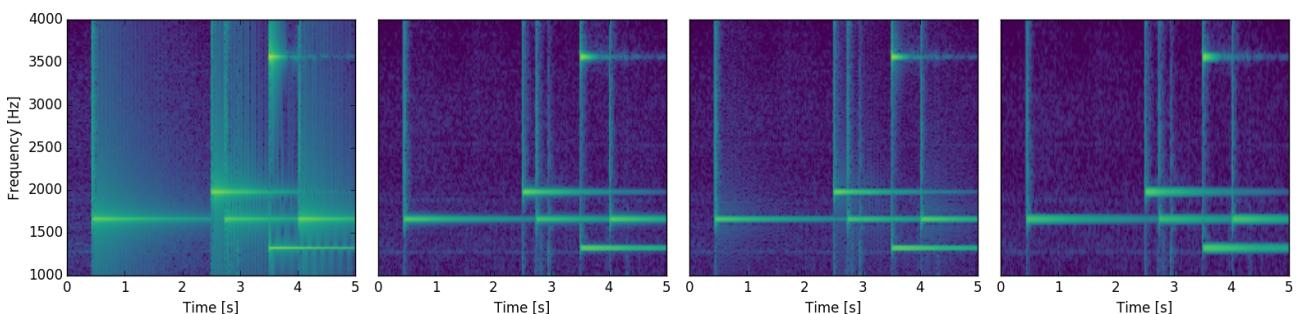
```



Сл. 4.6: Спектрограма на аудиозаписот Zvona2.wav⁵ добиен со употреба на Хамингов прозорец со должина 2048 одбирачи (46 ms)

※ **Важно!** Најважните два параметри во пресметување на спектрограмот се:

1. **Типот на прозорец.** Разликите во спектрограмот кои се добиваат со различните прозорци може да се воочат во деталите прикажани на Сл. 4.7.
2. **Големината на прозорецот.** Колку е поголем прозорецот толку повеќе точки, односно поголема резолуција, во спектарот на рамката земена од сигналот. Но исто така, колку поголем прозорец толку погруба временска резолуција на STFT анализата, Сл. 4.8. Имено, временските моменти на промена во спектарот биваат размачкани. Така, за добра временска резолуција ни требаат пократки прозорци (рамки од сигналот), кои пак повлекуваат лоша фреквенциска резолуција. Потребата од компромис меѓу време и фреквенција е главниот недостаток кај спектрограмите, односно STFT анализата.

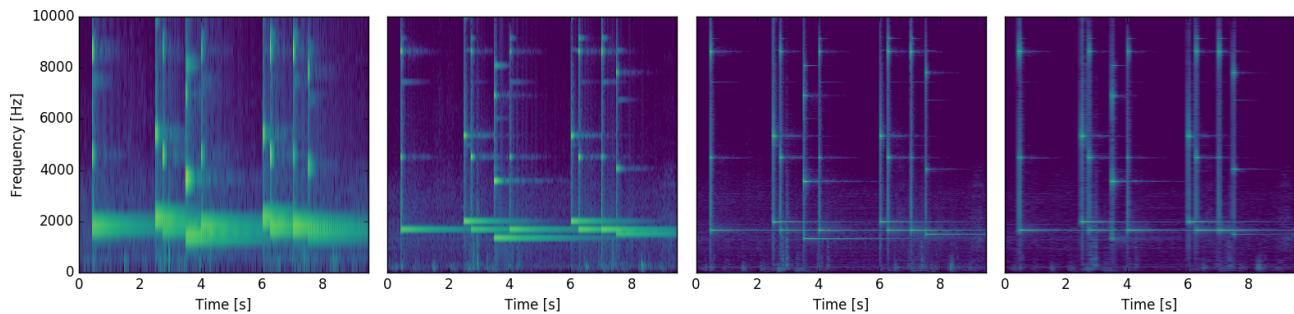


Сл. 4.7: Детали од спектрограмот на аудиозаписот Zvona2.wav добиени за различни типови на прозорци: правоаголен, Ханов, Хамингов и Блекман-Харисов, сите со должина 2048 одбирачи.

§ Дополнително. За прикажување на амплитудата на спектрограмот постојат најразлични низи на бои, наречени мапи на боја. Долго време во стандардна употреба е мапата *jet*, но таа има низа на недостатоци, поради кои денес се исфрла од употреба. Мапата искористена за приказ на спектрограмите во ова поглавје е новата *viridis* која е претставена на конференцијата SciPy 2015⁶.

⁵Звукот Gentle Glockenspiel од bbatv е земен од Freesound.org. <http://freesound.org/people/bbatv/sounds/332932/>

⁶Особено интересното претставување на новата мапа може да го погледнете на следниот линк: Nathaniel Smith and Stéfan van der Walt – A Better Default Colormap for Matplotlib <https://www.youtube.com/watch?v=xAoljeRJ3lU>



Сл. 4.8: Спектрограми на аудиозаписот Zvona2.wav добиени со употреба на Хамингов прозорец со различни должини: 128, 512, 4096 и 8192 одбираоци.

✓ **Задача за час.** Со помош на спектрограмот да се илустрира ефектот на преклопување (aliasing) преку намалување на фреквенцијата на семплирање без нископропусно филтрирање на аудиосигналот.

Поглавје 5

Филтри

Филтрите претставуваат системи чија примарна намена е обликувањето на спектарот на аудиосигналите. Тие се нераздвоен дел од дигиталното аудио. Аналогни и дигитални филтри се потребни за ограничување на спектарот во AD конверзијата и за обликување на излезниот аналоген сигнал во DA конверзијата; дигитални филтри имаме долж каналот за пренос на дигиталното аудио (трансмисија, дигитално снимање), а се основни градбени единки на еквилајзерите. Аналогните филтри се реализираат со помош на збир на пасивни и активни електронски елементи, чија нагоденост е неопходна, но захтевна и скапа. Од друга страна, реализацијата сложени филтерски структури во дигитален домен е многу поекономична, а нумеричките операции на кои се базираат дигиталните филтри не се подложни на временски и температурни влијанија.

5.1 Основи на дигиталните филтри

Дигиталните филтри претставуваат **линеарни и временски инваријантни (ЛВИ)**, односно **линеарни и инваријантни на поместување дискретни системи**¹. Овие две особини изискуваат ако за даден влезен сигнал $x[n]$ системот го дава излезниот сигнал $y[n] = H\{x[n]\}$, тогаш мора да важи (Богданова, 1997):

$$H\left\{\sum_{m=1}^M a_k x_k[n]\right\} = \sum_{m=1}^M a_k y_k[n], \quad \forall a_k \text{ -- линеарност и} \quad (5.1)$$

$$H\{x[n-k]\} = y[n-k], \quad \forall k \text{ -- инваријантност на поместување.} \quad (5.2)$$

Секој ЛВИ систем е во потполност описан од неговиот **импулсен одсив** $h[n]$ односно неговата **преносна функција** во z -домен $H(z)$, или пак конечно од неговата **фрејквенциска преносна функција** во Фурьеов домен $H(\omega)$. Како и за останатите ЛВИ системи, излезниот сигнал на дигиталните филтри $y[n]$ за даден влезен сигнал $x[n]$ може да се добие во трите домени со следните релации (Rabiner and Schafer, 1978):

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m]h[n-m], \quad (5.3)$$

$$Y(z) = H(z)X(z), \quad (5.4)$$

$$Y(\omega) = H(\omega)X(\omega). \quad (5.5)$$

За системот да биде остварлив, нужно е тој да биде **каузален**, односно да важи:

$$h[n] \equiv 0, \quad n < 0. \quad (5.6)$$

¹Linear time-invariant (LTI), односно linear shift-invariant (LSI) системи.

За системот пак да биде стабилен потребен и доволен услов е да важи:

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty. \quad (5.7)$$

Според импулсниот одсив филтрите ги делиме на две основни групи и тоа филтри со:

- **конечен импулсен одсив (FIR²)** и
- **бесконечен импулсен одсив (IIR³).**

FIR филтрите имаат одредени предности над IIR филтрите, а тоа се пред сè нивната стабилност и нивната линеарна фазна карактеристика. IIR филтрите пак можат да постигнат подобра амплитудна фреквенциска карактеристика за помал ред на филтерот.

Сите ЛВИ системи кои се практично применливи како дигитални филтри можат да се описат со **диференцната равенка**:

$$y[n] = \sum_{i=0}^p b_i x[n-i] - \sum_{i=1}^q a_i y[n-i]. \quad (5.8)$$

Според ова равенство секој одбирок на излезниот сигнал $y[n]$ зависи од p претходни примероци на влезниот сигнал и q минати примероци од излезниот, односно во системот има повратна врска, па велиме дека тој е **рекурзивен**. IIR филтрите секогаш се реализираат со повратна врска, од каде произлегуваат и проблемите со нивната стабилност. FIR филтрите можат да бидат реализирани и со повратна врска, но најчесто се без неа. Преносната функција на системот описан од диференцната равенка можеме да ја добиеме од (5.8):

$$y[n] + \sum_{i=1}^q a_i y[n-i] = \sum_{i=0}^p b_i x[n-i], \quad (5.9)$$

$$\left(1 + \sum_{i=1}^q a_i z^{-i}\right) Y[z] = \sum_{i=0}^p b_i z^{-i} X(z), \quad (5.10)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^p b_i z^{-i}}{1 + \sum_{i=1}^q a_i z^{-i}}. \quad (5.11)$$

Тука имплицитно претпоставуваме дека $a_0 = 1$. Во практата овој услов секогаш се обезбедува преку нормализирање на останатите a коефициенти. Бројот на коефициенти во повратната врска на IIR филтрите вообичаено се зема да е еднаков со бројот на коефициенти во директната врска. Па, важи:

$$q = p = N, \quad (5.12)$$

каде со N се означува **редот на филтерот**. Преносната карактеристика $H(z)$ исто така може да биде претставена преу нејзините **полови** и **нули** во z -рамнината:

$$H(z) = \frac{A \prod_{i=1}^p (1 - c_i z^{-1})}{\prod_{i=1}^q (1 - d_i z^{-1})}. \quad (5.13)$$

Фреквенциската карактеристика на филтрите ја добиваме директно од преносната карактеристика со замената $z = e^{j\omega}$:

$$H(\omega) = \frac{\sum_{i=0}^p b_i e^{-ji\omega}}{1 + \sum_{i=1}^q a_i e^{-ji\omega}}. \quad (5.14)$$

²Finite impulse response

³Infinite impulse response

Таа може дополнително да се изрази преку нејзината амплитуда и фаза како:

$$H(\omega) = |H(\omega)|e^{j\angle H(\omega)} = A(\omega)e^{j\phi(\omega)}, \quad (5.15)$$

каде $A(\omega)$ е амплитудната **фреквенциска карактеристика** на филтерот, додека $\phi(\omega)$ е неговата **фазна карактеристика**. Според амплитудната карактеристика разликуваме пет типови на филтри и тоа:

- **нископропусни,**
- **високопропусни,**
- **пропусни на опсег,**
- **непропусни на опсег и**
- **сепропусни.**⁴

Кога фазната карактеристика на филтерот $\phi(\omega)$ е линеарна тогаш групното доцнење $\tau(\omega)$ е константно:

$$\tau(\omega) = -\frac{d\phi(\omega)}{d\omega} = \text{const.} \quad (5.16)$$

Ова значи дека филтерот нема да внесе фазни изобличувања. Со тоа, компонентите на сигналот на различни фреквенции нема да бидат различно задоцнети па ќе биде задржана нивната компактност во излезниот сигнал, односно нема да дојде до нивно расејување. Строго линеарна фазна карактеристика може да се добие само со FIR филтри и тоа само кога нивниот импулсен одсив е симетричен во однос на средниот примерок $h[\frac{N-1}{2}]$:

$$h[n] = h[N - 1 - n], \quad (5.17)$$

или пак кога е антисиметричен во однос на него:

$$h[n] = -h[N - 1 - n]. \quad (5.18)$$

Тука редот на филтерот N ја дава и должината на импулсниот одсив, што не е случај кај IIR филтрите.

Строго линеарната фазна карактеристика не може да биде постигнато со IIR или со аналогните филтри (Богданова, 1997). Разликуваме четири типа на FIR филтри со линеарна фазна карактеристика:

Тип I – Симетричен импулсен одсив, N непарен

Импулсниот одсив на овој тип на филтри можеме да го запишеме како:

$$h[n] = h[0]\delta[n] + h[1]\delta[n - 1] + \cdots + h[\frac{N-1}{2}]\delta[n - \frac{N-1}{2}] + \cdots \quad (5.19)$$

$$\cdots + h[1]\delta[n - (N-2)] + h[0]\delta[n - (N-1)], \quad (5.20)$$

па за преносната функција имаме:

$$H(z) = h[\frac{N-1}{2}]z^{-\frac{N-1}{2}} + \sum_{i=0}^{\frac{N-3}{2}} h[i] \left(z^{-i} + z^{-(N-1-i)} \right) \quad (5.21)$$

$$= z^{-\frac{N-1}{2}} \left(h[\frac{N-1}{2}] + \sum_{i=0}^{\frac{N-3}{2}} h[i] \left(z^{-i+\frac{N-1}{2}} + z^{i-\frac{N-1}{2}} \right) \right) \quad (5.22)$$

$$= z^{-\frac{N-1}{2}} \sum_{i=0}^{\frac{N-1}{2}} a[n] \frac{z^n + z^{-n}}{2}, \quad (5.23)$$

⁴Овој тип на филтри наоѓа примена кај системите за синтеза на дигитално ехо и реверберација.

каде:

$$a[n] = \begin{cases} h[\frac{N-1}{2}], & n = 0 \\ 2h[-n + \frac{N-1}{2}] & n = 1, 2, \dots, \frac{N-3}{2} \end{cases} \quad (5.24)$$

Од (5.23) може да се пресмета фреквенциската карактеристика на филтерот:

$$H(\omega) = e^{-j\omega\frac{N-1}{2}} \sum_{n=0}^{\frac{N-1}{2}} a[n] \cos(n\omega). \quad (5.25)$$

Тип II – Симетричен импулсен одсив, N парен

Следејќи ја истата постапка како за FIR филтерот од тип I можеме да дојдеме до фреквенциската карактеристика на типот II:

$$H(\omega) = e^{-j\omega\frac{N-1}{2}} \sum_{n=1}^{\frac{N-1}{2}} b[n] \cos((n - \frac{1}{2})\omega), \quad (5.26)$$

каде:

$$b[n] = 2h[-n + \frac{N}{2}], \quad n = 1, 2, \dots, \frac{N}{2}. \quad (5.27)$$

Од (5.26) може да се види дека без оглед на вредноста на коефициентите на филтерот $b[n]$ неговата фреквенциска карактеристика ќе биде 0 за $\omega = \pm\pi$. Поради тоа, овој тип на филтер не може да се употреби како нископропусен или пропусник на опсег.

Тип III – Антисиметричен импулсен одсив, N непарен

Фреквенциската карактеристика на типот III е:

$$H(\omega) = e^{-j(\omega\frac{N-1}{2} - \frac{\pi}{2})} \sum_{n=1}^{\frac{N-1}{2}} c[n] \sin(n\omega), \quad (5.28)$$

каде:

$$c[n] = 2h[-n + \frac{N-1}{2}], \quad n = 1, 2, \dots, \frac{N-1}{2}. \quad (5.29)$$

Од (5.28) следи дека без оглед на вредноста на $c[n]$ неговата фреквенциска карактеристика ќе биде 0 за $\omega = 0$ и за $\omega = \pm\pi$. Поради тоа, овој тип на филтер може да се употреби само како пропусник на опсег.

Тип IV – Антисиметричен импулсен одсив, N парен

Фреквенциската карактеристика на овој филтер е:

$$H(\omega) = e^{-j\omega\frac{N-1}{2}} \sum_{n=1}^{\frac{N}{2}} d[n] \sin((n - \frac{1}{2})\omega), \quad (5.30)$$

каде:

$$d[n] = 2h[-n + \frac{N}{2}], \quad n = 1, 2, \dots, \frac{N}{2}. \quad (5.31)$$

Од (5.30) следи дека фреквенциска карактеристика ќе биде 0 за $\omega = 0$, па овој тип на филтер не може да се употреби како нископропусен.

Постојат различни пристапи за **дизајн на дигитални филтри**. Трите најпознати методи за дизајн на FIR филтри се:

- **метода на прозорци,**

- дизајн заснован на DFT,
- оптимален дизајн на еднаквобранести филтри.

Од друга страна за дизајн на IIR филтри најпознати методи се:

- Батерворт,
- Бесел,
- Чебишев,
- Елиптичен.

5.2 Дизајн на FIR филтер

Дизајнот на FIR филтри со методата на прозорци се заснова на апроксимација на идеалната фреквенциска карактеристика на филтерот преку земање на прозорец од импулсниот одсив кој одговара на неа. Ќе ја илустрираме методата за нископропусен филтер со гранична фреквенција ω_l . Идеалната фреквенциска карактеристика на ваков филтер би била:

$$H_l(\omega) = \begin{cases} 1, & |\omega| \leq \omega_l \\ 0, & \omega_l < |\omega| \leq \pi \end{cases}. \quad (5.32)$$

Импулсниот одсив на овој идеален филтер $h_l[n]$ ќе биде:

$$h_l[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_l(\omega) e^{j n \omega} d\omega = \frac{1}{2\pi} \int_{-\omega_l}^{\omega_l} e^{j n \omega} d\omega = \frac{\sin(n\omega_l)}{\pi n}. \quad (5.33)$$

Овој импулсен одсив не може практично да се реализира поради тоа што неговото траење е бесконечно и тој не е каузален. За таа цел во методата на прозорци се зема само дел од него преку негово множење со избран прозорец. И тука важат истите дискусиии во Поглавјата 4.3.2 и 4.3.3 за влијанието на спектралните карактеристики на прозорците и компромисот помеѓу домените време и фреквенција.

За да ја илустрираме оваа метода ќе напишеме код кој идеалниот филтер ќе го конструира во Фурьеов домен во $Nfft = fs$ точки, неговиот импулсен одсив ќе го пресмета со употреба на IDFT, за од него да задржиме еден дел со употреба на правоаголен прозорец со должина N . Поради тоа што импулсниот одсив го пресметуваме од идеалната фреквенциска карактеристика, имплементираниот алгоритам за дизајн на FIR филтер всушност претставува комбинација од методите за дизајн базирани на DFT и примената на прозорци.⁵

⁵Благодарност за предочување на ова доц. д-р Јелена Ќертиќ, професор по дигитално процесирање на сигнали на Електротехничкиот Факултет при Универзитетот во Белград.

```
# -*- coding: utf-8 -*-
from __future__ import division
import numpy as np
from matplotlib import pyplot as plt
from math import pi
from scipy.io import wavfile
import das
from scipy import fftpack as fp
from scipy import signal as sig

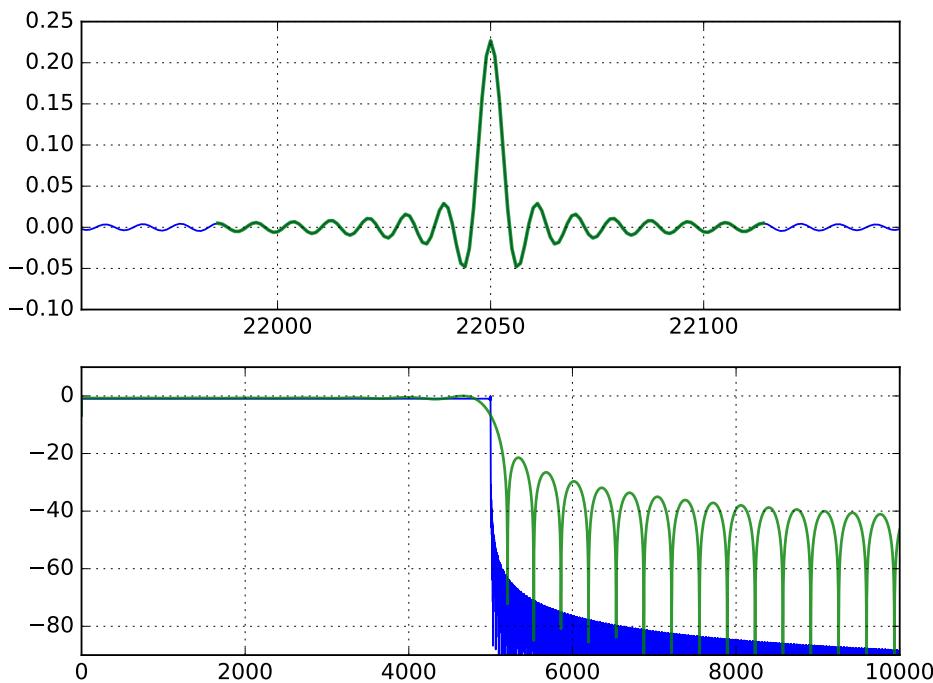
# конструкција на филтерот
fs = 44100
w_l = 5000 / (fs/2) # гранична фреквенција нормализирана до 1
Nfft = fs
w = np.linspace(0,pi,Nfft/2+1)
H_l = np.zeros(f.size)
H_l[w < w_l * pi] = 1 # идеална карактеристика 0 до  $\pi$ 
H_l = np.append(H_l, H_l[-2:0:-1]) # огледална слика  $\pi$  до  $2\pi$ 
h_l = fp.ifft(H_l, Nfft)
h_l = fp.fftshift(h_l) # го правиме системот каузален
```

```
%%% метода на прозорци
N = 128 + 1 # должина на прозорецот = ред на филтерот
Nh = (N-1)/2
Nffth = Nfft/2

win = sig.get_window('boxcar', N) # правоаголен прозорец
n_win = np.arange(Nffth-Nh,Nffth+Nh+1) # индекси кои ни требаат
h_rect = h_l[tuple(n_win),] * win # реален филтер

%%% плотирање
plt.figure()
plt.subplot(211)
plt.plot(h_l, linewidth=1, alpha=1)
plt.plot(n_win, h_rect, linewidth=2, alpha=.8)
plt.grid()
plt.subplot(212)
f, H_l = das.get_spectrum(h_l, fs)
H_l = H_l - np.max(H_l) # нормализација
plt.plot(f, H_l)
f, H_rect = das.get_spectrum(h_rect, fs)
H_rect = H_rect - np.max(H_rect) # нормализација
plt.plot(f, H_rect, linewidth=1.5, alpha=.8)
plt.grid()
```

Конструираната фреквенциска карактеристика на идеалниот нископропусен филтер и онаа добиена со методата на прозорци со употреба на правоаголен прозорец се прикажани заедно со нивните импулсни одсиви во Сл. 5.1. Може да се види деградацијата на фреквенциската карактеристика кај добиениот нископропусен филтер поради ограничување на идеалниот импулсен одсив. Исто така може да се види ефектот на множење со правоаголниот прозорец во временски домен, кое претставува конволуција на фреквенциските карактеристики на двата сигнали во спектрален домен.



Сл. 5.1: Импулсни одсиви и фреквенциски карактеристики на конструираниот идеален нископропусен филтер и оној добиен со методата на прозорци.

§ Дополнително. На Сл. 5.1 може да се види дека ни идеалниот импулсен одсив ја нема оригинално конструираната идеална фреквенциска карактеристика. Ова е поради тоа што вистински идеалниот импулсен одсив има бесконечно многу примероци, а овој кој ние го конструирајме има f_s . Всушност и тој самиот како да сме го добиле со методата на прозорци од вистинскиот.

5.3 Споредба на карактеристиките на FIR и IIR филтри

5.4 Еквализација

Еквализацијата настанала како потреба да се израмни фреквенцискиот одсив на преносните системи. Звукот низ својот пат од изворот до слушателот поминува низа аудио уреди кои со своите неидеалности го изобличуваат неговиот првобитен спектар. Така, микрофоните немаат идеално рамна фреквенциска карактеристика – нивната чувствителност е помалка на ниските фреквенции како и на многу високите. Сличен е и одсивот на звучниците. Ова „обојување“ на звукот може соодветно да се компензира со употреба на уред со инверзна фреквенциска карактеристика на онаа на каналот. Со ова преносната функција на склопот ја поништува нерамномерноста на каналот т.е. ефективно го исправа истиот. Од тука доаѓа името на процесот еквализација, односно на уредот – еквализатор.

Во обработката на аудио (аналогна и дигитална), во употреба е поширока дефиниција за еквализацијата. Според неа, еквализацијата е процес кој го обликува спектарот на аудио сигналот (како било кој аналоген филтер). Еден пример за практична примена на еквализацијата е истакнување на ритамот во дискотеките преку засилување на ниските фреквенции на музиката. Од друга страна засилувањето на високите фреквенции во корист на ниските навидум му дава на звукот поголема гласност иако целокупната негова моќност останува иста. Ова го користат маркетинг агенциите за да го привлечат вниманието на слушателите.

Основниот начин на реализација на еден еквализаторот се состои од примена на повеќе филтри

пропусници на опсег, вскладени да го препокријат целиот звучен опсег. На тој начин, тие го делат на подопсези кои можат да се истакнат или потиснат преку одредување на засилувањето на филтерот. Важно е вкупната фазна карактеристика на филтрите да биде линеарна. Мора да се води сметка и за доцнењата кои филтрите ги внесуваат, ако тие меѓусебно се разликуваат, тогаш резултантниот сигнал ќе има изразени изобличувања.

Во аналогната техника бројот на филтри е ограничен со цената на уредот, додека во дигиталната техника со процесирачката моќ. Затоа добрите аналогни аудио системи најчесто вклучуваат едноставна bass-treble еквализација, додека дигитални аудио плеери вообичаено поддржуваат 6 и повеќе-канална еквализација. Притоа дигиталните филтри се најчесто од IIR тип од 4 ред, за намалување на комплексноста. Притоа, треба да се обрне особено внимание на нелинеарната фазна карактеристика на IIR филтрите.

Еквализаторот во Python ќе го направиме преку паралелна врска на 3 филтри:

- филтер за бас – нископропусен до 400 Hz,
- филтер за средни – пропусник на опсег од 400 до 4000 Hz и
- филтер за високи фреквенции – високопропусен над 4 kHz.

```

1 #%% дефинирање на филтрите
2 N = 7 # ред на филтрите
3 # bass
4 f_b = 400 / (fs/2)
5 b_b, a_b = sig.iirfilter(N, f_b, btype='low', ftype='butter')
6 # mid
7 f_ml = 400 / (fs/2)
8 f_mh = 4000 / (fs/2)
9 b_m, a_m = sig.iirfilter(N, [f_ml, f_mh], btype='band', ftype='butter')
10 # treble
11 f_h = 4000 / (fs/2)
12 b_t, a_t = sig.iirfilter(N, f_h, btype='high', ftype='butter')
```

```

13 # преносни функции
14 w, H_b = sig.freqz(b_b,a_b)
15 f = w /pi * (fs/2)
16 H_b = 20*np.log10(H_b)
17 w, H_m = sig.freqz(b_m,a_m)
18 H_m = 20*np.log10(H_m)
19 w, H_t = sig.freqz(b_t,a_t)
20 H_t = 20*np.log10(H_t)

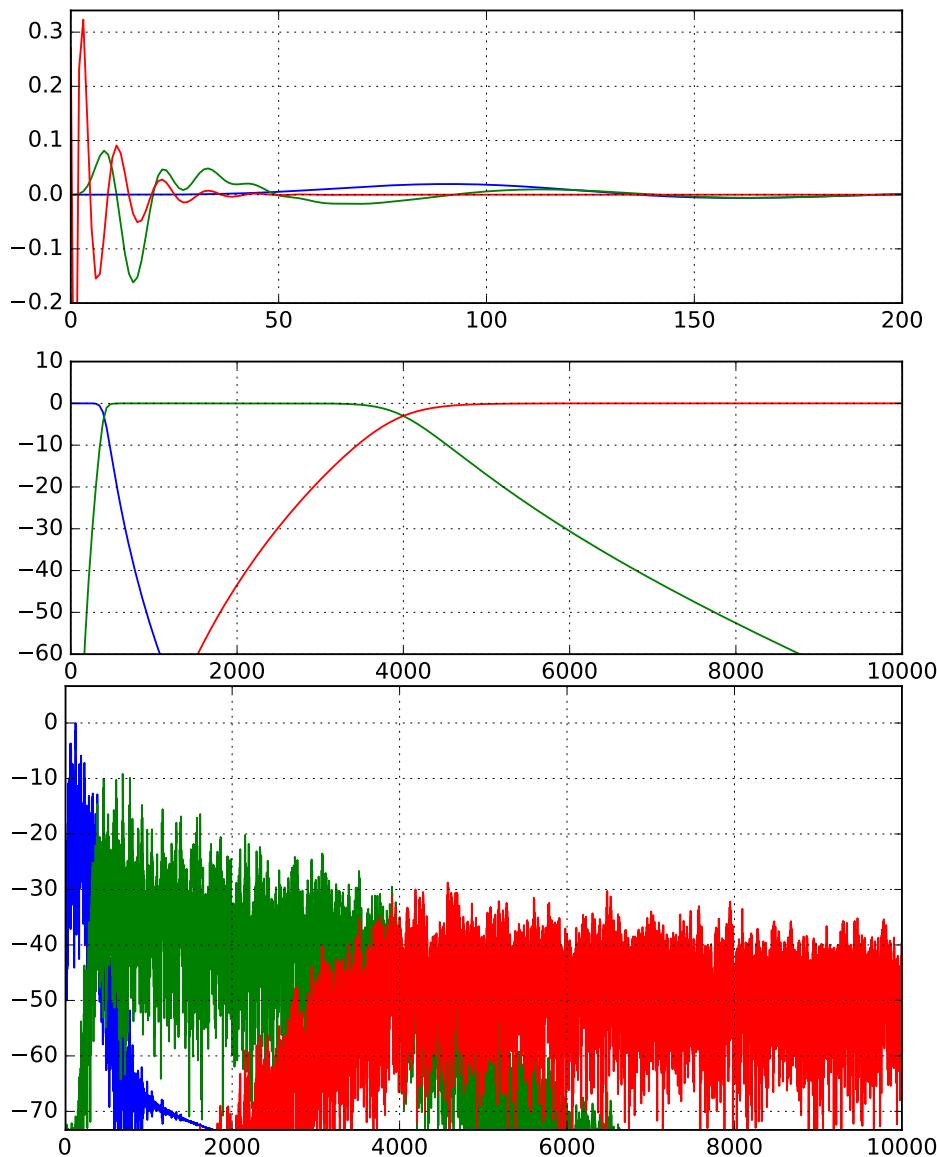
21
22 %% плотирање
23 plt.figure()
24 plt.subplot(211)
25 x = np.zeros(1000)
26 x[0] = 1
27 h_b = sig.lfilter(b_b, a_b, x)
28 h_m = sig.lfilter(b_m, a_m, x)
29 h_t = sig.lfilter(b_t, a_t, x)
30 plt.plot(h_b)
31 plt.plot(h_m)
32 plt.plot(h_t)
33 plt.axis([0, 200, -.2, .34])
34 plt.grid()
35 plt.subplot(212)
36 plt.plot(f, H_b)
37 plt.plot(f, H_m)
38 plt.plot(f, H_t)
39 plt.axis([0, 10000, -60, 10])
40 plt.grid()

41
42 %% филтрирање
43 fs, wav = wavfile.read('audio/disco_inferno.wav')
44 wav_bass = sig.lfilter(b_b,a_b, wav)
45 wav_mid = sig.lfilter(b_m,a_m, wav)
46 wav_treble = sig.lfilter(b_t,a_t, wav)

47
48 %% засилување
49 g_bass = -20 # во dB
50 g_mid = 0 # во dB
51 g_treble = 20 # во dB
52 g_b = 10**(g_bass/20)
53 g_m = 10**(g_mid/20)
54 g_t = 10**(g_treble/20)
55 wav_out = g_b*wav_bass + g_m*wav_mid + g_t*wav_treble

56
57 %% плотирање
58 f, wav_spec = das.get_spectrum(wav, fs)
59 f, wav_bass_spec = das.get_spectrum(wav_bass, fs)
60 f, wav_mid_spec = das.get_spectrum(wav_mid, fs)
61 f, wav_treble_spec = das.get_spectrum(wav_treble, fs)
62 plt.figure()
63 plt.plot(f, wav_spec)
64 plt.plot(f, wav_bass_spec)
65 plt.plot(f, wav_mid_spec)
66 plt.plot(f, wav_treble_spec)
67 plt.grid()

```



Сл. 5.2: Импулсни одсиви и фреквенциски карактеристики на трите IIR филтри од дизајнираниот еквализатор и добиени подопсези од аудиосигналот.

```

68 #%% преслушување
69 import os
70 wav_out = wav_out / np.max(np.abs(wav_out))
71 wavfile.write('audio/disco_inferno_eql.wav', fs,
72                 np.array(wav_out*2**15, dtype='int16'))
73 os.system('play audio/disco_inferno.wav')
74 os.system('play audio/disco_inferno_eql.wav')

```

Импулсните одсиви, фреквенциските карактеристики на трите дизајнирани филтри и добиените подопсези од аудиосигналот се прикажани на Сл. 5.2.

5.5 Филтри непропусни на фреквенција – notch филтри

Една од примените на дигиталните филтри за која неприкосновени се IIR филтрите е потиснувањето на одредена фреквенција во аудиосигналите. Најчесто потребата за ваков тип на процесирање се јавува кога треба да се потисне брумот од градската мрежа на 50 Hz, кој се

нарекува **брум**, кој влегол во аудиосигналот пред неговата дигитализација. За таа цел потребно е филтерот непропусник да има што потесен опсег и да биде со што поголемо slabeeње, односно голем Q-фактор. Овие типови на филтри се нарекуваат **notch филтри**.

5.6 Дигитални аудиоэффекти базирани на филтри

Најпознатите дигитални аудиоэффекти базирани на филтри се **wah-wah** и **фејзерот**. Wah-wah ефектот⁶ е првенствено направен за изведба на музика на електрична гитара, иако за првпат го пронашле трубачите и тромбонистите во 1920^{te}. Тој се изведува со помош на филтер пропусник на опсег чија централна фреквенција се менува со време а под контрола на свирачот преку потенциометар поставен во педала за дозирање. Аудиосигналот кој филтерот го дава на излез се меша со оригиналниот сигнал за да се добие крајниот ефект. Во електронската музика педалата може да биде заменета од нискофреквенциски осцилатор (НФО).

Фејзерот⁷ исто така претставува гитарски ефект добиен со употреба на каскада на notch филтри чии што фреквенции на потиснување периодично се менуваат. На овој начин тие вршат селективно slabeeње на делови од спектарот на сигналот кое може да се види како траг во спектограмот. Повторно менувањето на фреквенцијата на овие филтри може да биде направена од свирачот преку педала со потенциометар или со употреба на НФО. Дополнително фејзерот може да се реализира и со сепропусни филтри дискутирани во поглавјето 6.1.

⁶Wikipedia: Wah-wah (music). https://en.wikipedia.org/wiki/Wah-wah_%28music%29

⁷Wikipedia: Phaser (effect). https://en.wikipedia.org/wiki/Phaser_%28effect%29

Поглавје 6

Процесирање на аудиосигналите базирано на доцнење

Голем број на аудиоэффекти се базираат на генерирање на задоцнети верзии од аудиосигналот и нивно додавање на оригиналниот сигнал. Двата основни аудиоэффекти кои се реализираат на овој начин се **ехото** и **реверберацијата**. Пред појавата на дигиталното аудио, овие ефекти биле правени со електро-механички склопови, некои од нив големи колку и цела просторија. Во дигитален домен тие се генерираат со хардверска, а почесто софтверска, обработка на сигналите.

6.1 Основи на процесирање со задоцнување

Одсивот на систем кој генерира задоцната верзија на аудиосигналот и истата ја додава на него може да го претставиме со помош на следната диференцна равенка:

$$y[n] = x[n] + b_D x[n - D]. \quad (6.1)$$

Може да се види дека оваа диференцна равенка претставува специјален случај на општата диференцна равенка (5.8). Тука D е доцнењето на сигналот во број на примероци, b_D е неговото слабеење, а земено е дека $b_0 = 1$. Импулсниот одсив и преносната функција на овој систем се дадени со:

$$h[n] = 1 + b_D \delta[n - D], \quad (6.2)$$

$$H(z) = \frac{Y(z)}{X(z)} = 1 + b_D z^{-D}. \quad (6.3)$$

Поради конструктивното и деструктивно собирање на задоцната верзија од аудиосигналот со неговиот оригинал за различни фреквенции, фреквенциската карактеристика на системот ќе има низа максимуми и минимуми распоредени на еднакво растојание. Поради овој облик овие системи се нарекуваат уште и **чешлести филтри**. Ваквата фреквенциска карактеристика е неповолна за обработка на аудиосигналите поради спектралните изобличувања кои ќе бидат внесени во сигналот.

За реализација на повеќекратни задоцнети верзии од аудиосигналот постојат два можни пристапи. Едниот е да се додадат нови ненулти коефициенти во диференцната равенка на FIR филтерот дадена во (6.1). Имаме:

$$y[n] = x[n] + b_{D_0} x[n - D_0] + b_{D_1} x[n - D_1] + \dots + b_{D_{M-1}} x[n - D_{M-1}], \quad (6.4)$$

$$h[n] = 1 + b_{D_0} \delta[n - D_0] + b_{D_1} \delta[n - D_1] + \dots + b_{D_{M-1}} \delta[n - D_{M-1}], \quad (6.5)$$

$$H(z) = \frac{Y(z)}{X(z)} = 1 + b_{D_0} z^{-D_0} + b_{D_1} z^{-D_1} + \dots + b_{D_{M-1}} z^{-D_{M-1}} = 1 + \sum_{m=0}^{M-1} b_{D_m} z^{-D_m}. \quad (6.6)$$

Тука, со M е означен бројот на задоцнети верзии додадени во сигналот одредени со нивните доцнења D_m и коефициенти на слабеење b_{D_m} .

Вториот пристап се базира на употреба на IIR филтри кои по својата природа имаат бескрајно долг импулсен одсив а со тоа и повеќекратни задоцнети верзии на сигналот кои се распоредени на мултикли од доцнето $m \times D$ на првото доцнене. Слабеењето на овие задоцнети верзии претставува степен од слабеењето на првото a_D^m .

$$y[n] = x[n] - a_D y[n - D] \quad (6.7)$$

$$h[n] = \frac{1}{1 + a_D \delta[n - D]} \quad (6.8)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 + a_D z^{-D}} \quad (6.9)$$

Како и претходно фреквенциската карактеристика на овој IIR филтер е од чешлест облик, па внесува изобличувања во излезниот аудиосигнал. Овојпат, максимумите се на местата на минимумите кај FIR филтерот и обратно, минимумите кај IIR филтерот се на локациите на максимумите на FIR филтерот.

Последниот податок можеме да го искористиме за дизајн на систем кој би генерирал задоцнети верзии од аудиосигналот а притоа би имал рамна фреквенциска карактеристика која не би внесувала изобличувања во излезниот сигнал. Ова може да се направи преку едноставна комбинација на комплементарните FIR и IIR системи дадена со следните равенства:

$$y[n] = b_D x[n] + x[n - D] - b_D y[n - D], \quad (6.10)$$

$$h[n] = \frac{b_D + \delta[n - D]}{1 + b_D \delta[n - D]}, \quad (6.11)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_D + z^{-D}}{1 + b_D z^{-D}}. \quad (6.12)$$

Ваквиот систем и покрај генерирањето на повеќекратни еха има амплитудна фреквенциска карактеристика еднаква на 1, поради што се нарекува **филтер сепропусник**. Филтрите сепропусници се користат во генерирањето на најразлични аудиоэффекти базирани на доцнење.

6.2 Exo

Ехото претставува ефект во кој на аудиосигналот му се додаваат една или повеќе задоцнети верзии од него самиот, при што меѓу тие задоцнети верзии и оригиналниот сигнал може да се направи јасна дистинкција. Во случај кога имаме повеќе задоцнети верзии велиме дека се работи за **повеќекратното echo**. За реализација на ехото ќе ги искористиме FIR и IIR филтрите кои ги разгледавме, како и филтрите сепропусници на опсег. При употреба на IIR филтер или сепропусник можеме да кажеме дека системот генерира **бескрајно echo**.

Најпрвин да ги увеземе потребните модули и пакети.

```

1 # -*- coding: utf-8 -*-
2 from __future__ import division
3 import numpy as np
4 from matplotlib import pyplot as plt
5 from math import pi
6 from scipy.io import wavfile
7 from scipy import signal as sig
8 import copy as cp
9 import os

```

Сера да ги имплементираме овие три типови на системи.

```

10 fs = 22050
11
12 # FIR exo
13 Dt = 0.5 # sec
14 D = int(Dt*fs) # samples
15 b_D = 0.5
16 b_fir = np.zeros(D+1)
17 b_fir[0] = 1
18 b_fir[D] = b_D
19
20 # повеќекратно FIR exo
21 D0t = 0.2
22 D0 = int(D0t*fs) # samples
23 D1t = 0.3
24 D1 = int(D1t*fs) # samples
25 b_fir_mul = cp.copy(b_fir)
26 b_fir_mul[D0] = 0.4
27 b_fir_mul[D1] = 0.3
28
29 # бескрајно exo
30 a_iir = np.zeros(D+1)
31 a_iir[0] = 1
32 a_iir[D] = b_D
33
34 # сепропусник
35 b_ap = np.zeros(D+1)
36 b_ap[0] = b_D
37 b_ap[D] = 1
38 a_ap = cp.copy(a_iir)

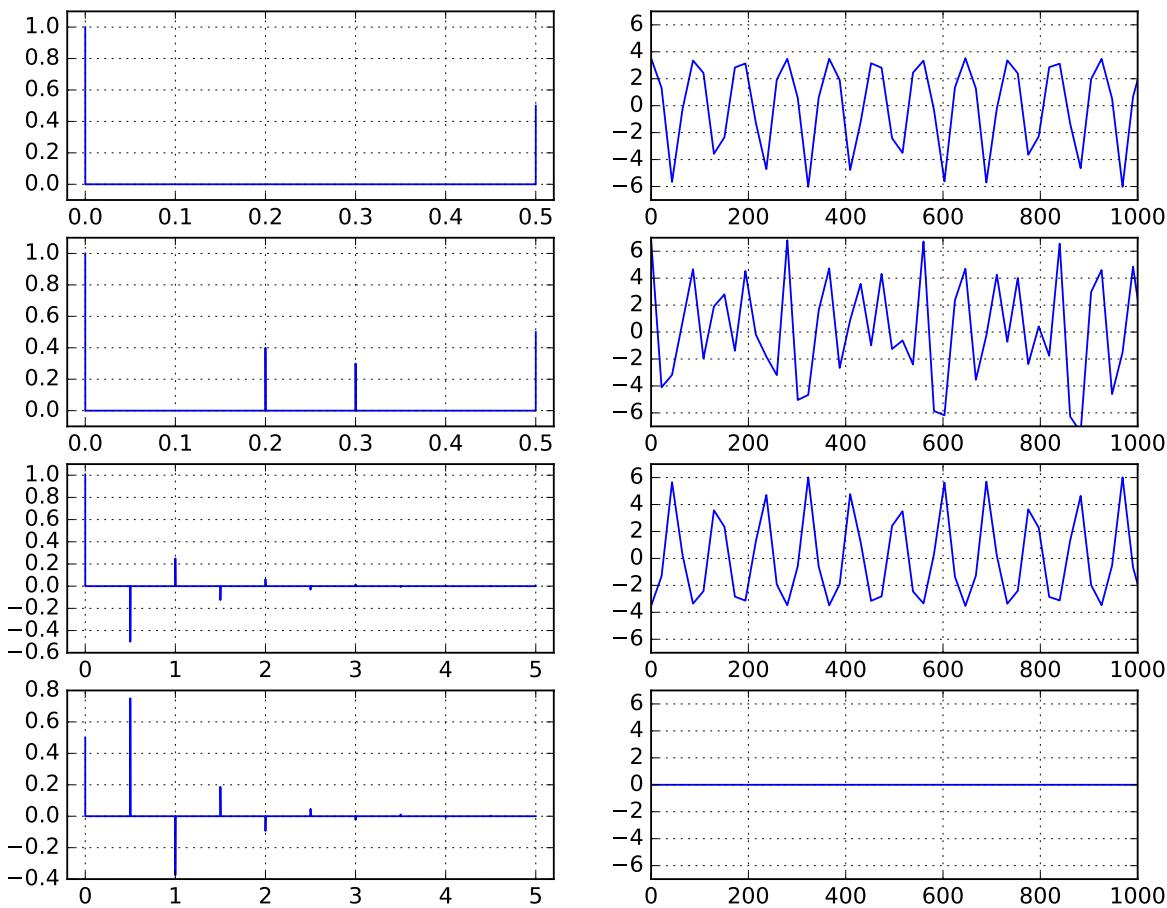
```

Следно ќе ги пресметаме импулсните и фреквенциските одсиви на овие четири системи.

```

39 # FIR
40 w, H_fir = sig.freqz(b_fir, [1])
41 f = w / pi * fs/2
42 H_fir = 20*np.log10(np.abs(H_fir))
43
44 # FIR multiple
45 w, H_fir_mul = sig.freqz(b_fir_mul, [1])
46 H_fir_mul = 20*np.log10(np.abs(H_fir_mul))
47
48 # IIR
49 x = np.zeros(5*fs)
50 x[0] = 1
51 h_iir = sig.lfilter([1], a_iir, x)
52 w, H_iir = sig.freqz([1], a_iir)
53 H_iir = 20*np.log10(np.abs(H_iir))
54
55 # AP
56 h_ap = sig.lfilter(b_ap, a_ap, x)
57 w, H_ap = sig.freqz(b_ap, a_ap)
58 H_ap = 20*np.log10(np.abs(H_ap))

```



Сл. 6.1: Импулсни и фреквенциски одсиви на четирите системи за генерирање на ехо (од горе надолу): FIR филтер – единично ехо, FIR филтер – повеќекратно ехо, IIR филтер – бесконечно ехо и филтер сепропусник.

Пресметаните импулсни и фреквенциски одсиви на четирите системи за генерирање на ехо се претставени на Сл. 6.1. Може да се види дека првите три системи навистина имаат чешлеста амплитудна фреквенциска карактеристика, додека филтерот сепропусник има рамна карактеристика со амплитуда од 0 dB.

Конечно, да ги искористиме имплементираните системи за процесирање на еден аудиосигнал и да ги чуеме резултатите.

```

59 #%% филтрирање
60 fs, wav = wavfile.read('audio/Pato_22K.wav')
61 wav_echo = sig.lfilter(b_fir, [1], wav)
62 wav_echo = sig.lfilter(b_fir_mul, [1], wav)
63 wav_echo_iir = sig.lfilter([1], a_iir, wav)
64 wav_echo_ap = sig.lfilter(b_ap, a_ap, wav)
65
66 #%% преслушување
67 os.system('play audio/Pato_22K.wav')
68 wavfile.write('audio/Pato_echo_fir.wav', fs, np.array(wav_echo, dtype='int16'))
69 os.system('play audio/Pato_echo_fir.wav')
70 wavfile.write('audio/Pato_echo_fir_mul.wav', fs, np.array(wav_echo, dtype='int16'))
71 os.system('play audio/Pato_echo_fir_mul.wav')
72 wavfile.write('audio/Pato_echo_iir.wav', fs, np.array(wav_echo, dtype='int16'))
73 os.system('play audio/Pato_echo_iir.wav')
74 wavfile.write('audio/Pato_echo_ap.wav', fs, np.array(wav_echo, dtype='int16'))
75 os.system('play audio/Pato_echo_ap.wav')
```

6.3 Реверберација

Реверберацијата или **ревербот** претставува ефект во кој се генерираат и наддодаваат многу повеќе задоцнети верзии од аудиосигналот со што се постигнува нивно аудиторно слевање во еден здружен оддек. Реверберацијата е таа која му дава просторност на звукот, односно преку нејзе можеме да заклучиме во каков вид на просторија го слушаме или е снимен звучниот сигнал. Реверберацијата може да се реализира со едноставно паралелно или сериско надоврзување на повеќе системи за генерирање на еднократно или повеќекратно ехо со густо распоредени доцнења.

6.4 Други дигитални ефекти базирани на доцнење

Други аудиоэффекти базирани на употреба на дицнење или сепропусни филтри се **хор** и **фленџ**. Ефектот **хор**¹ е ефект во кој на аудиосигналот му се додаваат една или повеќе негови верзии добиени со променливо но мало задоцнување. На тој начин се постигнува впечаток дека наместо еден музички извор, во аудиосигналот постојат повеќе извори кои се релативно добро усогласени, но на моменти приметно разгодени.

Фленџ² ефектот се добива преку примена на FIR систем за еднократно ехо кое има променливо доцнење. Аудитивно овој ефект е сличен на фејзерот описан во поглавјето 5.6, кој пак може да се реализира со каскада од променливи сепропусни филтри. Разликата меѓу двата ефекта е во тоа што фленџот се базира на чешлестиот облик на фреквенциската карактеристика на FIR филтерот, па генерира максимуми и минимуми во излезниот сигнал кои се во хармониски сооднос, односно се мултипли од основниот максимум. Од друга страна, ова не е случај кај фејзерот.

¹Wikipedia: Chorus effect. https://en.wikipedia.org/wiki/Chorus_effect

²Wikipedia: Flanging. <https://en.wikipedia.org/wiki/Flanging>

Поглавје 7

Компресија на аудиосигналите

Поради големиот габарит на чистото ИКМ аудио, а од друга страна ограничениот мемориски простор на персоналните компјутери, односно ограничениот пропусен опсег на телекомуникациските врски за вмрежување, се пристапува кон компресија на WAV датотеките. Постојат различни методи со кои може да се изврши компресијата на дигиталното аудио (Spanias et al., 2006). Во зависност од методот на компресија дефинирани се и различни формати на компресирано дигитално аудио. Постојат две основни групи на методи за компресија:

- методи без загуби „lossless“ и
- методи со загуби „lossy“.

Методите за компресија без загуби не исфрлаат никакви информации од оригиналното аудио. Овие методи не се многу различни од општите методи за компресија и успеваат да ја намалат големината на звучните записи на 30 – 80% од онаа на оригиналната датотека, во зависност од нејзината содржина. Така, говорот трпи поголема компресија отколку музиката. Најупотребуваниот формат од овој тип е **Free Lossless Audio Codec (FLAC)**¹ кој претставува отворен кодек. Други формати од оваа категорија се Shorten, Monkey’s Audio, ATRAC Advanced Lossless, Apple Lossless, WMA Lossless, TTA, WavPack итн.

Методите со загуби ја намалува големината на дигиталниот запис жртвувајќи дел од неговиот квалитет. Најчесто тој дел човековото уво и не може да го чуе поради различни акустички феномени, а во најголема мера поради маскирањето. Тоа се прави со употреба на психоакустички модели и исфрлање на непотребните информации од звучниот запис (Painter and Spanias, 2000). На тој начин тие постигнуваат смалување на големината на аудио датотеките 10, па и повеќе пати од големината на некомпресираната датотека. Така, стандардната mp3 (MPEG-1 Layer 3) компресија го намалува битскиот проток од номиналните 1411 kb/s кај аудио-CD-то до 128 kb/s, а при тоа да се зачува субјективното чувство за еднаков квалитет кај слушателот. Други познати методи за компресија на аудио со загуби се слободниот **Vorbis Ogg (OGG)**², како и затворените Windows Media Audio (WMA) и Advanced Audio Coding (AAC). Квалитетот на кодираното аудио според направените двојни слеп тестови е речиси еднаков кај четирите формати, при среден проток од 128 kb/s, додека OGG форматот покажува најдобри перформанси при нискиprotoци (под 64 kb/s) и при повисоки protoци (180 kb/s). Vorbis форматот е и единствениот непатентиран отворен метод за компресија. Со истекување на патентнот на Техниколор за mp3 во САД на 16. април 2017, mp3 се придржува на слободните стандарди за компресија.³

¹Free Lossless Audio Codec (FLAC). <https://xiph.org/flac/>

²Vorbis audio compression. <https://xiph.org/vorbis/>

³Off.net.mk – Многу поважната сторија околу „смртта на mp3“ што беше пропуштена. <http://off.net.mk/vesti/tehnologija/mnogu-povazhnata-storija-okolu-smrtta-na-mp3-shto-beshe-propushtena>

7.1 MPEG-1 ниво III

Како еден од најраспространетите претставници на групата алгоритми за компресија на аудио со загуби ќе ги разгледаме основите на MPEG-1 ниво III, односно mp3 алгоритмот. Тој е главниот стандард за пренос и зачувување на компресирано аудио, како на интернет мрежата, така и на персоналните компјутери, во преносливите мултимедијални уреди итн. Иако денес постојат поразвиени алгоритми за компресија на аудиото, главните придобивки кои тие ги носат се за ниските битски брзини, т.е. за поголемите степени на компресија. Над 128 kbit/s квалитетот на компресија со mp3 стандардот е на доволно високо ниво и по денешните стандарди, па тој сеуште го задржува приматот во овој домен.

MPEG (Moving Pictures Experts Group) е експертска група чиишто главни задачи се:

- да објавува технички резултати и извештаи поврзани со компресија на аудио и видео,
- да одреди начин за мултиплексирање на видео, аудио и информациски протоци во единствен проток,
- да даде описи и синтакса за алатки за компресија на аудио и видео до ниски протоци за Интернет апликации и апликации кои работат со ограничен опсег.

MPEG стандардите не даваат точни спецификации за реализација на кодерот, туку го дефинираат типот на информациите кои тој треба да ги даде, како и начинот на кој декодерот треба да ги протолкува при декомпресијата. До сега се објавени 5 различни MPEG стандарди поврзани со дигиталното аудио:

- MPEG-1,
- MPEG-2 BC (Backwards-Compatible),
- MPEG-2 NBC/AAC (Non-Backward Compatible/Advanced Audio Coding),
- MPEG-4 и
- MPEG-7,
- MPEG-21.

Од нив последните два не се стандарди за компресија. MPEG-7 дефинира дескриптори на аудио/видео содржините, со чија помош може да се опишат за побрз пристап до нив во бази на податоци. MPEG-21 дефинира мултимедијална рамка и нуди менаџирање и заштита на интелектуална сопственост.

Два различни термини се поврзани со MPEG стандардите, тоа се: фаза и ниво. Фазите одговараат на главниот тип на MPEG аудио стандардот: MPEG-1, MPEG-2, MPEG-4 итн. Нивоата означуваат фамилии на алгоритми за кодирање внатре во MPEG фазата. Нивоа се дефинирани само во MPEG-1 и MPEG-2 и тоа:

- MPEG-1 ниво-I, -II и -III,
- MPEG-2 ниво-I, -II и -III.

MPEG-1 Аудио (ISO/IEC 11172-3) стандардот е првиот аудио стандард, објавен од MPEG групата во 1992, по четиригодишна напорна работа на усогласено истражување на светските експерти од областа на аудио компресијата. Неговата намена била да се овозможи стерео-CD-квалитет. MPEG-1 подржува стерео аудио CD квалитет на 192 kbit/s. Тоа го достигнува со примена на флексибилна техника за хибриден компресија на аудиото која се базира на сплет од неколку методи и тоа:

- подопсежно разлагање,
- анализа со банки на филтри,

- психоакустична анализа,
- адаптивна сегментација,
- трансформациско кодирање,
- динамично доделување на битови,
- не-униформна квантизација и
- ентрописко кодирање.

MPEG-1 аудио кодекот работи со 16-битен ИКМ влез со fs од 32, 44,1 и 48 kHz. Тој нуди различни модови на работа за моно, стерео, двојно моно и заедничко (joint) стерео. Протоците на компресираното аудио се дефинирани во опсег 32 – 192 kbit/s за моно и 64 – 384 kbit/s за стерео.

MPEG-1 архитектурата содржи три нивоа со растечка комплексност, доцнење и квалитет на компресириот сигнал. Секое повисоко ниво ги вклучува функционалните блокови од претходните.

Во нивото II, влезниот сигнал се разложува на 32 критично подсемплирани подопсези со употреба на **банка на филтри** од типот PQMF (Pseudo Quadrature Mirror Filter). Каналите се еднакво распоредени на тој начин што влезен сигнал со фреквенција на семплирање од 48 kHz се разлага на подопсези од по 750 Hz, а подопсезите се децимирани (подсемплирани) со однос 32:1. Прототипниот филтер е од 511^{ви} ред така што вкупната дисторзија, карактеристична за PQMF банките на филтри, останува под прагот на чујноста, а слабеење во непорпусниот дел од спектарот од 96 dB осигурува занемарливо меѓуопсежко влијание.

Целта на **психоакустичката анализа** е определување на **праговите на приметлива дисторзија JND** (Just Noticeable Distortion) за различните подопсези. Тие ја одредуваат максималната грешка на квантизација која може да си ја дозволи кодерот при распределбата на расположливиот број на битови помеѓу подопсезите. При динамичкото доделување на битови, приоритет (највеќе битови) ќе добијат подопсезите со најниски JND. Нивната вредност зависи од два параметри прикажани на Сл. 7.1:

- **прагот на чујност** – со кој е определена минималната спектрална амплитуда која човек може да ја чуе,
- **фреквенциското маскирање** – со кое се одредува делот од амплитудниот спектар кој воопшто не може да биде чуен поради присуството на изразени спектрални компоненти.⁴

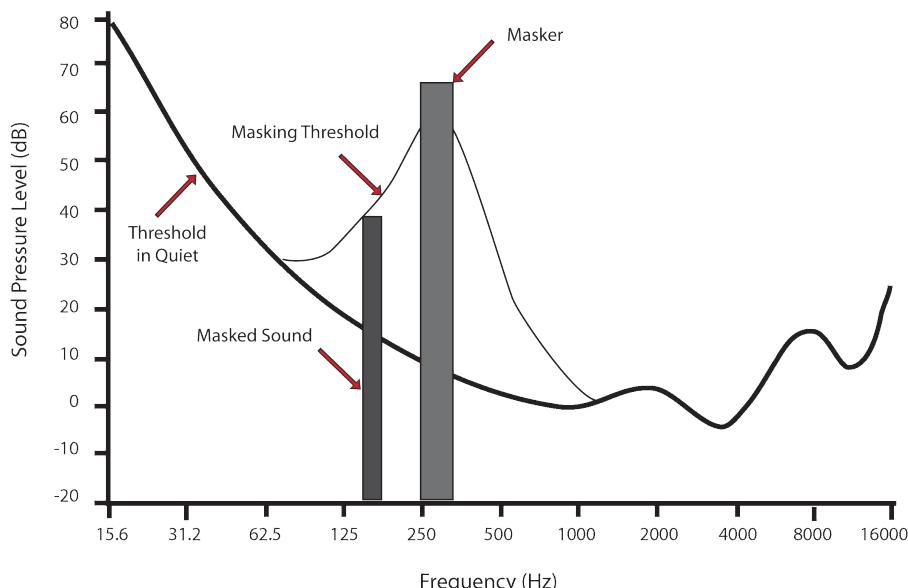
Притоа фреквенциското маскирање е тесно поврзано со **критичните опсези**⁵ на аудиторниот систем кои го одредуваат опсегот на фреквенции кои можат да бидат маскирани од една компонента во зависност од нејзината фреквенција. Тие всушност ја даваат ширината на пропусниот опсег на базилијарната мембра на функцијата од фреквенцијата.

За пресметување на JND се употребува FFT во 512 (ниво I) односно 1024 (ниво II) точки. На секој подопсег се врши динамичка компресија во блокови од по 12 примероци, со што максималната амплитуда на децимираните примероци во секој блок е 1. При тоа за секој блок се дефинира коефициент на размер (scale factor), со кој се врши нормализацијата. Секој блок соодветствува на $12 \cdot 32 = 384$ влезни примероци, односно 8,7 ms на 44,1 kHz.

На крајот започнува итеративна процедура на доделување на битови која ги користи пресметаните JND прагови за секој подопсег за одбирање на оптимален квантизер за тој подопсег (од дадено почетно множество на квантизери). Изборот на квантизерите се прави така што двата

⁴Wikipedia: Psychoacoustics – Masking effects: https://en.wikipedia.org/wiki/Psychoacoustics#Masking_effects

⁵Wikipedia: Critical band: https://en.wikipedia.org/wiki/Critical_band



Сл. 7.1: Прагот на чујност и појавата на фреквенциско маскирање на кои се базира принципот на работа на психоакустичката аудиокомпресија.⁶

критериуми – зададениот краен проток и пресметаниот JND, се задоволени. Коефициентите на размер се квантанизираат со 6 битови за секој подопсег, а изборот на квантизерот со 4 битови.

Во нивото I, децимираните подопсежни секвенци се квантанизираат и испраќаат на приемникот проследени со квантизираните коефициенти на размер и избраните квантизери. Во нивото II пак:

- психоакустичкиот модел има поголема FFT резолуција,
- максимална резолуција на подопсежните квантизери е зголемена на 16 битови, а понизок вкупен проток се остварува со намалување на бројот на понудени квантизери за повисоките подопсези,
- количината на додатна информација за коефициентите на размер е намалена со искористување на временското маскирање. Ова се прави преку разгледување на особините на 3 соседни блокови од 12 примероци и се испраќаат 1, 2 или 3 коефициенти заедно со 2-битен додатен параметар кој ја означува нивната поврзаност.

MPEG ниво-III алгоритмот работи врз последователни рамки на податоци. Секоја се состои од 1152 примероци аудио; секоја рамка понатаму се дели на две подрамки од по 576 примероци, наречени гранули. Декодерот може да ја декодира секоја гранула посебно.

Во трет е воведена хибридна банка на филтри се употребува за зголемена фреквенциска резолуција и подобра апроксимација на критичните опсези на човековото уво. Исто така, хибридната банка на филтри вклучува адаптивна сегментација за подобрување на контрола врз пред-ехото. Хибридната банка на филтри е конструирана со надоврзување на секој подопсежен филтер на блок за адаптивна MDCT (Modified Discrete Cosine Transform). На тој начин, за разлика од нивоата I и II, во нивото III не се кодираат филтрирани сегменти на аудио, туку нивните коефициенти во **DCT домен**. DCT транформацијата е позната по својата способност на компактирање на енергијата на сигналите, односно претставување на голем дел од нивната енергијата со мал број на коефициенти. Поради ова, таа често се користи за нивна компресија, на пример кај JPEG стандардот.

На крајот, нивото III користи софистицирано доделување на битови и стратегии на квантизација кои се базираат на:

⁶By Audio_Mask_Graph.jpg: Daxx4434derivative work: Cradle (talk) - Audio_Mask_Graph.jpg, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=8390519>

- неуниформна квантација,
- анализа преку синтеза и
- ентрописко кодирање.

Доделувањето на битови и квантацијата на MDCT спектралните линии се изведува преку вгнездена јамка која употребува и неуниформна квантација и **Хофманово кодирање**. Внатрешната јамка го прилагодува чекорот на квантација на трансформациските коефициенти за секој блок се додека не се задоволи зададениот битски проток. Надворешната јамка го контролира квалитетот на кодираниот сигнал преку анализа со синтеза, во однос на нивото на квантизацискиот шум спореден со пресметаните JND прагови.

7.2 Компресија на говор

Линеарното предиктивно кодирање – LPC (Linear Predictive Coding) е еден од најважните концепти во кодирањето на дигиталниот говор. Техниката овозможува високо-квалитетно кодирање со ниски битски протоци од редот на 2,4 kb/s. Таа е во употреба во многу телекомуникациски системи за пренос на говор, од кои најважен е GSM (Groupe Spécial Mobile) системот за мобилна комуникација.

LPC се базира на **моделот извор-филтер** на создавање на говорот прикажан на Сл. 7.2. Во овој модел изворот го претставуваат неколку модули кои ги моделираат побудните механизми на човековиот говорен апарат и тоа:

- **генератор на поворка на импулси** – ја моделира периодичната побуда од гласилките,
- **генератор на шум** – ги модулира турбуленциите во воздушниот проток предизвикани од стеснувања во вокалниот тракт при изговор на согласките,
- **засилување** – ја моделира активноста на белите дробови кои го генерираат субглоталниот притисок кој претставува побуда на целиот систем.

Дополнително во изворот постои и преклопник за селекција на моменталниот тип на побуда според тоа дали гласот кој се изговара е звучен или беззвучен.

Филтерот во моделот извор-филтер ја моделира преносната функција на вокалниот тракт која одговара на промените во напречниот пресек предизвикани од поставеноста на **артикулаторите**: јазикот, мекото непце, вилицата, и усните. Тој ја моделира оваа функција преку IIR филтер кој содржи само полови:

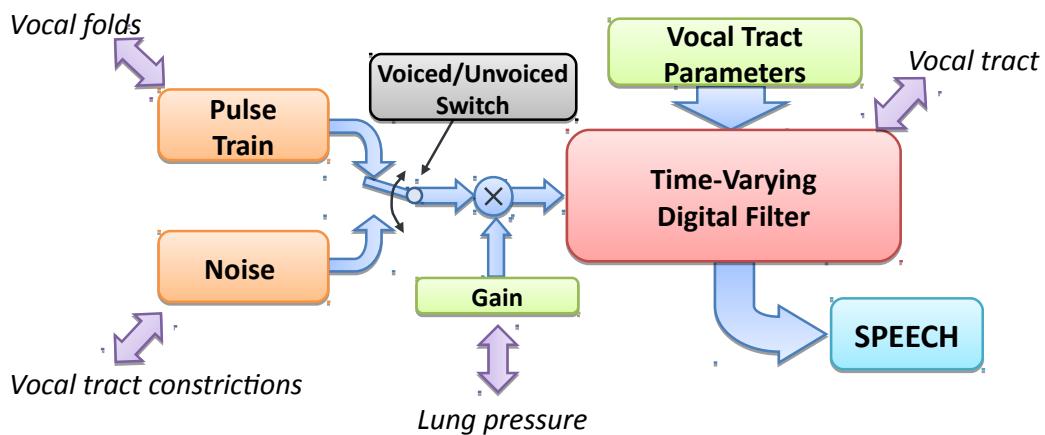
$$H(z) = \frac{Y(z)}{X(z)} = \frac{G}{1 - \sum_{k=1}^p a_k z^{-k}}, \quad (7.1)$$

каде $H(z)$ е преносната функција на филтерот во z -домен, $Y(z)$ е излезниот говорен сигнал, $X(z)$ е побудниот сигнал кој е добиен од изворот во моделот, G е засилувањето кое иако претставува дел од изворот, е интегрирано во филтерот, a_k се коефициентите на филтерот, а p е неговиот ред. Во временски домен ова равенство би било:

$$y[n] = \sum_{k=1}^p a_k y[n-k] + Gx[n]. \quad (7.2)$$

Со употреба на моделот извор-филтер, говорот може да се пренесува/зачувува без употреба на аудио примероци. На овој начин, може да се генерира доволно блиска апроксимација на говорниот сигнал, по цена на голема заштеда во простор. Параметрите кои се екстрахираат за секоја рамка од говорниот сигнал се:

- **звукочност** – дали се работи за звучен или беззвучен глас,



Сл. 7.2: Моделот извор-филтер на принципот на создавање на говорот.

- **засилување** – енергијата на сигналот во рамката,
- **коефициенти на филтерот** – кои соодветствуваат на преносната карактеристика на вокалниот тракт, и
- **висина** – периода на основниот хармоник.

Најосновната имплементација на LPC кодерот нуди одлична разбираливост на кодираниот говор. Но, кавлитетот на излезниот говор не е на високо ниво и има карakterистичен роботски призвук. **Federal Standard (FS) 1015** кодекот од 1982 г. е првиот кодер базиран на LPC. Протокот кој тој го остварува е 2,4 kbit/s, што во споредба со вообичаениот проток на дигиталниот говор во телефонијата од 64 kbit/s⁷, претставува компресија од 26 пати. Бидејќи филтерот кој го моделира вокалниот тракт е со ред 10, кодекот се нарекува и **LPC-10**. Ако не го знаете моделот, параметрите кои се испраќаат, немаат никакво значење. Поради тоа, FS 1015 кодекот е направен за американската војска, за потоа да го присвои и НАТО.

И покрај одличната разбираливост, овој кодер пати од лош, синтетички квалитет на излезниот говор. Ова се должи на главниот недостаток на овој пристап – едноставноста на употребениот модел. Нефлексибилноста на изворот, кој може да работи исклучиво во еден од двата мода, претставува пречка во моделирањето на гласови од мешан тип, на пр. звучни согласки како „з“ /z/ или „в“ /v/, како и на меѓу-гласовни премини. Овој недостаток е надминат кај **CELP (Code-Excited Linear Prediction)** кодерот, кој на местото од едноставниот извор, работи со база на побуди поместени во една кодна книга. CELP моделот е оној кој е во употреба во GSM мрежата.

⁷8 kHz · 8 bit = 64 kbit/s

Поглавје 8

Процесирање на аудиосигналите базирано на линеарна предикција

Филтерот кој се користи во моделот извор-филтер, односно во компресијата на говорните аудиосигнали базирана на линеарна предикција (LPC) дискутирана во Поглавјето 7.2 го има обликот даден во (7.1):

$$H(z) = \frac{Y(z)}{X(z)} = \frac{G}{1 - \sum_{k=1}^p a_k z^{-k}}. \quad (8.1)$$

Од акустиката на вокалниот тракт пак, за верно претставување на назалните и фрикативите се потребни и нули во спектарот. Сепак се применува филтер кој содржи исклучиво полови затоа што коефициентите на вака дефинирирајќи филтер можат лесно да се одредат преку употреба на методот за **линеарна предиктивна анализа** врз говорниот сигнал. За апроксимација пак на нулите во преносната функција на вокалниот тракт, се земаат поголем број на полови во овој филтер. Линеарен предиктор со коефициенти α_k е дефиниран со следното равенство (Rabiner and Schafer, 1978):

$$\tilde{y}[n] = \sum_{k=1}^p \alpha_k y[n-k], \quad (8.2)$$

каде со $\tilde{y}[n]$ е означена предикцијата на сегашната вредност на излезниот сигнал базирана на тежинска сума на неговите p претходни примероци. Притоа грешката од предикција може да се пресмета како:

$$e[n] = y[n] - \tilde{y}[n] = y[n] - \sum_{k=1}^p \alpha_k y[n-k]. \quad (8.3)$$

Преносната функција на овој систем во z -домен е:

$$A(z) = 1 - \sum_{k=1}^p \alpha_k z^{-k}. \quad (8.4)$$

Споредувајќи ги овие равенства со (8.1) и (7.2):

$$y[n] = \sum_{k=1}^p a_k y[n-k] + Gx[n], \quad (8.5)$$

можеме да видиме дека ако е задоволен условот:

$$\alpha_k = a_k, \quad \text{за } k = 1, 2, \dots, p, \quad (8.6)$$

тогаш сигналот на грешка е еднаков на побудниот сигнал на моделот извор-филтер:

$$e[n] = Gx[n], \quad (8.7)$$

а преносната функција на филтерот на грешка $A(z)$ претставува инверзен филтер на $H(z)$:

$$H(z) = \frac{G}{A(z)}. \quad (8.8)$$

Основниот проблем во линеарната предикција претставува одредување на коефициентите α_k директно од говорниот сигнал за да се добие добра проценка на спектралните карактеристики на говорниот сигнал преку употреба на (8.8). Поради временската спектрална динамика на говорот, нужна е проценка на овие коефициенти за кратки отсекочи на сигналот добиени со методата на прозорци.

Основниот пристап во решавањето на овој проблем е одбирање на коефициенти на предикторот кои ќе ја минимизираат средната квадратна грешка на предикција. Постојат различни пристапи за решавање на овој проблем како методите со автокорелација и коваријанса. Најшироко применет алгоритам за одредување на коефициентите на предикторот е рекурзивниот метод на Дурбин (Rabiner and Schafer, 1978).

8.1 Анализа и синтеза на глас со линеарна предикција

За практично да се запознаеме со начинот на функционирање на линеарната предикција ќе ја илустрираме нејзината примена во синтеза на човечки глас. Тој процес е во с'ржта на нејзината примена за компресија на говорните сигнали. За целите на оваа анализа направете снимка од сопствениот глас како ги изговарате петте самогласки во македонскиот јазик. Тоа ќе го направиме со помош на Audacity.

8.1.1 Audacity

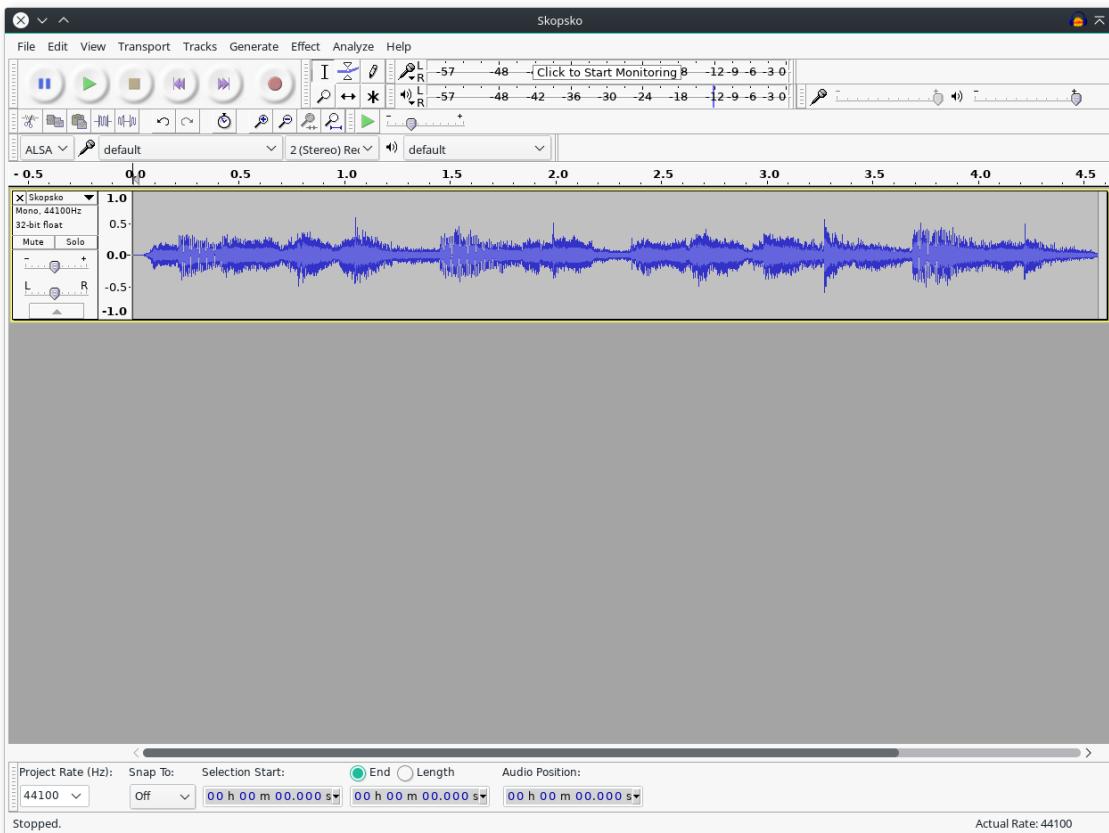
Audacity¹ е слободен софтвер за едитирање и снимање на дигитално аудио, достапен за сите оперативни системи, Сл. 8.1. Неговиот развој го започнале Доминик Мацони и Роџер Даненберг во 1999 во Универзитетот Карнеги Мелон и е иницијално објавен во 2000 како верзија 0.8. Од 2011, тој е 11-от најсимнуван софтвер на SourceForge, со 76,5 милиони симнувања. Audacity е добитник на наградата за најдобар проект за мултимедија од заедницата SourceForge 2007 и 2009. Во 2015 е преместен на FossHub каде за 4 месеци постигнува 10 милиони симнувања.²

Освен тоа што поддржува снимање од повеќе извори, Audacity може да се искористи за процесирање на сите типови на аудио, преку додавање на ефекти како нормализација, поткастрување, и прелевање (fading). Тој може да се користи за снимање и миксање на цели албуми, како што е случајот со групата Tune-Yards. Тој е во употреба и во националниот курс за ICT ниво 2 на OCR (Oxford, Cambridge and RSA Examinations) во Обединетото Кралство. Главните особини на Audacity вклучуваат:

- Вчитување и снимање на различни типови на аудио формати, како WAV, AIFF, MP3, Ogg Vorbis, FLAC, WMA, AAC, AMR и AC3.
- Снимање и репродукција на звук.
- Едитирање со неограничен број на undo.
- Автоматска поделба на аудио траки на дигитализирани снимки од касети или грамофонски плочи.
- Повеќеканално миксање.

¹Audacity. <http://www.audacityteam.org/>

²Wikipedia: Audacity (audio editor). [https://en.wikipedia.org/wiki/Audacity_\(audio_editor\)](https://en.wikipedia.org/wiki/Audacity_(audio_editor))



Сл. 8.1: Отворен аудио фајл во главниот прозорец на Audacity.

- Голем број на аудио ефекти и плагини. Додатни ефекти можат да се напишат во Nyquist кој е диалект на Lisp, а поддржани се плагини направени во отворениот LV2 стандард, како и VST плагини.
- Едитирање на амплитудната анвелопа.
- Намалување на шумот.
- Намалување на вокалите.
- Спектрална анализа со употреба на FFT.
- Подршка на повеќеканално дигитално аудио со фреквенција на семплирање до 96 kHz и резолуција до 32 bit.
- Прецизно нагодување на брзината на аудиото без промена во фреквенцијата на звукот.
- Нагодување на висината на тонот без промена на брзината.
- Можности за модерно повеќеканално едитирање.
- Работа на повеќе платформи.
- Приказ на ефектите базирани на LADSPA, VST (32-bit) и Audio Unit (OS X) во реално време.
- Зачувување и вчитување на кориснички пресети.

8.1.2 Моделирање на гласот /а/

Во нашата анализа ќе ја искористиме имплементацијата на линеарната предикција, поточно алгоритамот на Дурбин за линеарна предикција која е содржана во модулот `scikit.talkbox`³. Овој модул содржи некои основни функционалности за обработка на говор или за аудиосигналите поопшто и може едноставно да се инсталира користејќи го pip:

```
$ sudo pip install scikits.talkbox
```

Во кодов што следи ќе го вчитаме аудио записот со самогласката /а/ и ќе го прикажеме неговиот временски и спектрален облик, Сл. 8.2.

```
from __future__ import division
import numpy as np
from matplotlib import pyplot as plt
from scipy.io import wavfile
import os
from scipy import signal as sig
import das
from scikits.talkbox import lpc

#%%
# load wav
folder = 'audio/'
filename = 'glas_aaa.wav'
fs, wav = wavfile.read(folder+filename)
wav = wav / 2**15
t = np.arange(wav.shape[0]) / fs
wav = das.normalise(wav, 0)

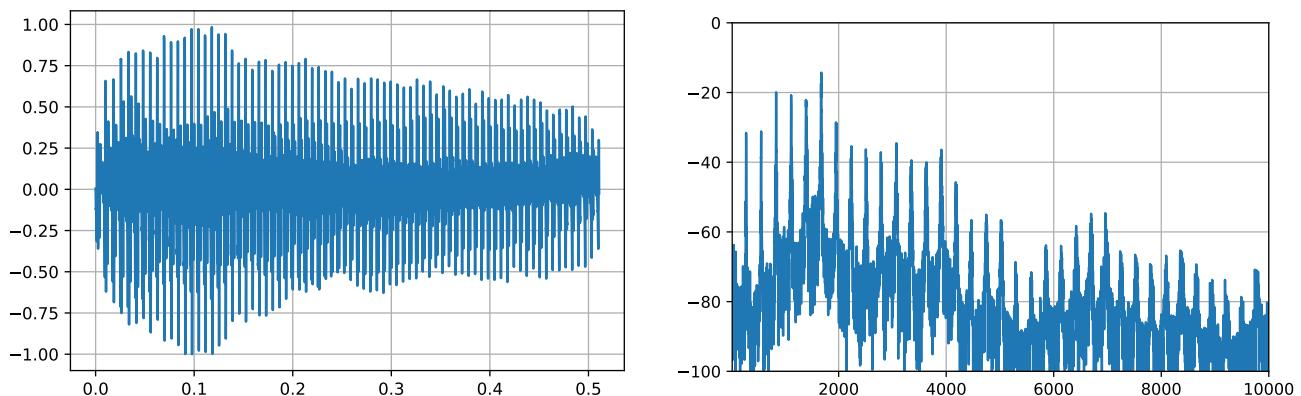
#%%
# plot time domain
plt.figure()
plt.plot(t, wav)
plt.grid('on')

#%%
# plot spectral domain
f, wav_spec = das.get_spectrum(fs, wav)
plt.figure()
plt.plot(f, wav_spec)
plt.xscale('log')
plt.grid('on')
plt.axis([0, 2e4, -100, 0])
```

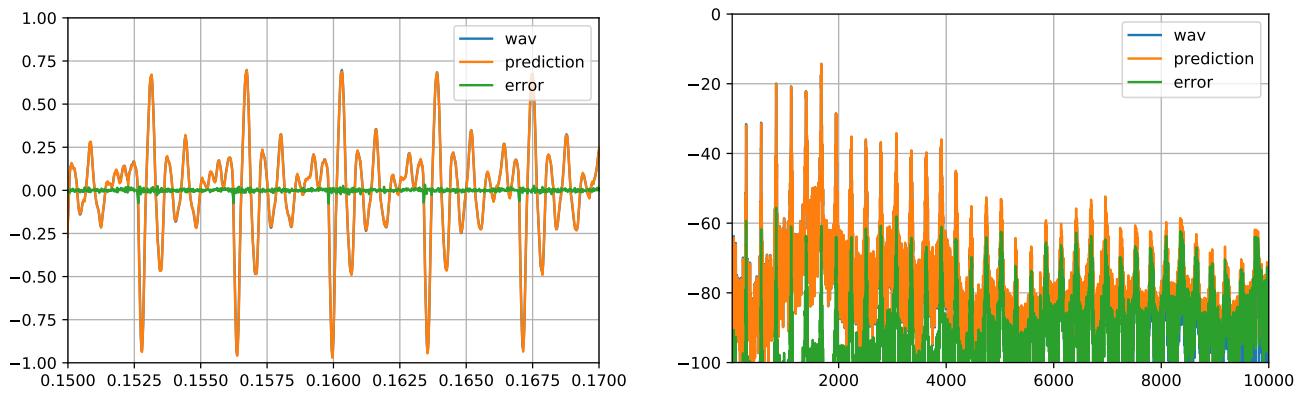
Во спектралниот облик на сигналот може да се видат хармониците кои се должат на периодичноста на сигналот, како и анвелопата на спектарот која има изразени врвови. Овие врвови одговараат на резонантните фреквенции на вокалниот тракт при изговор на гласот /а/. Кога се работи за самогласки ти се нарекуваат **форманти**. Сега ќе ги најдеме коефициентите на филтерот кои треба да ја опишат анвелопата на сигналот со употреба на функцијата `lpc`.

```
p = 24
a_filt, err, _ = lpc(wav, p)
b_inv = np.concatenate(([0], -a_filt[1:]))
wav_pred = sig.lfilter(b_inv, 1, wav)
wav_err = wav - wav_pred
```

³scikits.talkbox <http://www.scikits.appspot.com/talkbox>



Сл. 8.2: Приказ на временскиот (лево) и спектралниот (десно) облик на аудиосигналот на гласот /а/.



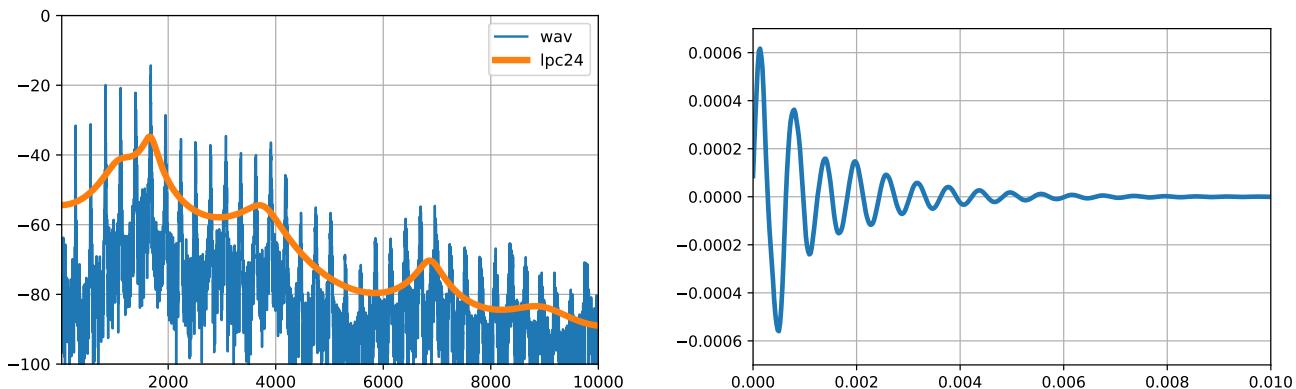
Сл. 8.3: Оригиналниот аудиосигнал, неговата предикција и сигналот на грешка во временски (лево) и во спектрален (десно) домен.

На Сл. 8.3 се претставени оригиналниот аудиосигнал, неговата предикција и сигналот на грешка. Може да се види дека разликата меѓу аудиосигналот и предикцијата добиена со филтер со ред 25 е многу мала и има мали импулси на почетокот на секоја периода. Ова е всушност сигналот кој алгоритамот за одредување на коефициентите за предикција го минимизира. Импулсите во него ги претставуваат моментите кога побудата на вокалниот тракт е максимална што одговара на моментите на **глотално затворање**, односно кога протокот на воздух низ гласилките постигнува брзина за која страничниот притисок не може да ги држи отворени па тие се затвораат. Оваа периодичност појасно се гледа во спектарот на сигналот на грешка кој ги содржи хармониците од оригиналниот сигнал. Уште поважно е што тој не ја содржи спектралната анвелопа, односно енергијата на сите негови хармоници е речиси иста. Може да се каже дека сигналот на грешка претставува спектрално „обелена“ верзија на оригиналниот сигнал.⁴ За да видиме како добиените коефициенти на предикторот ја опишуваат спектралната анвелопа на оригиналниот сигнал, ќе исцртаме фреквенциската карактеристика на добиениот филтер суперпонирана на спектарот на сигналот.

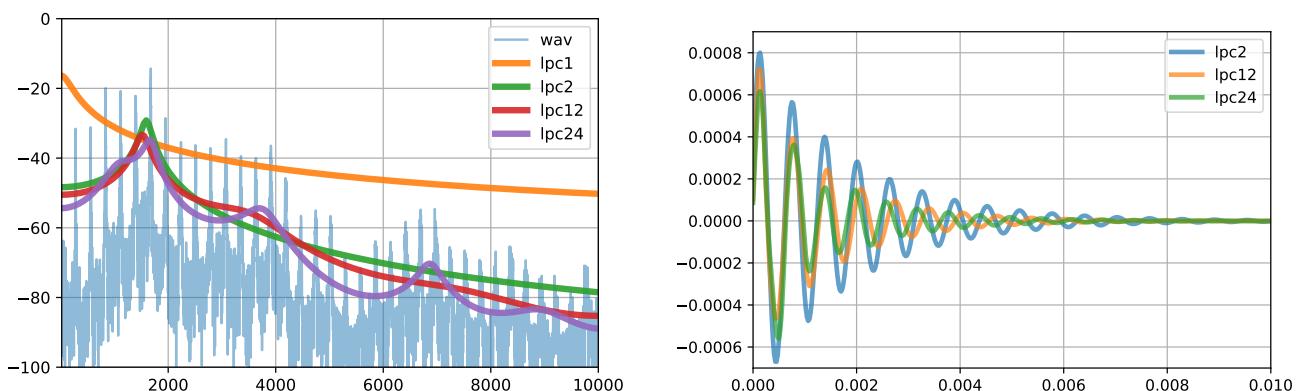
```
G = err
w, H_filt = sig.freqz(G, a_filt)
f_H = w / np.pi * fs/2
H_filt = 20*np.log10(np.abs(H_filt))
plt.figure()
plt.plot(f, wav_spec)
plt.plot(f_H, H_filt, lw=4)
```

Импулсниот одсив пак на филтерот ќе го добиеме со следниот код. Двата графици се прикажани на Сл. 8.4

⁴На англиски процесот на „белене“ на еден сигнал се нарекува „whitening“.



Сл. 8.4: Фреквенциската карактеристика на добиениот филтер за предикција од 25-ти ред (лево) и неговиот импулсен одсив (десно).



Сл. 8.5: Фреквенциски карактеристики на филтри за предикција од различен ред (лево) и нивните импулсни одсиви (десно).

```
excite = np.zeros(int(.050*fs))
excite[0] = 1
h_filt = sig.lfilter(G, a_filt, excite)
t_imp = np.arange(excite.size)/fs
plt.figure()
plt.plot(t_imp, h_filt, lw=3)
```

На Сл. 8.5 е дадена споредба на фреквенциските карактеристики и импулсните одсиви на филтри за линеарна предикција со различен ред. Може да се види дека филтер од прв ред го доловува само глобалниот пад на енергијата со растењето на фреквенцијата, додека филтерот од втор ред го фаќа главниот формант во спектралната анвелопа на гласот /а/. Како го зголемуваме бројот на коефициенти во филтерот така сè подобро тие ја претставуваат спектралната анвелопа на аудиосигналот. Едно непишано правило е дека при моделирање на говор бројот на коефициенти на филтерот треба да е еднаков на бројот на форманти, т.е. резонантни фреквенции, плус 2 пола за нулите. Вообичаено се зема дека во секој kHz од спектарот на сигналот има по еден формант, па од таму оптималниот ред е широчината на сигналот во kHz плус 2. Во нашиот случај за f_s од 44,1 kHz тоа значи дека оптималниот ред на филтерот би бил 22.

Литература

Ted Painter and Andreas Spanias. Perceptual coding of digital audio. *Proceedings of the IEEE*, 88(4): 451–515, 2000.

Lawrence R. Rabiner and Ronald W. Schafer. *Digital Processing of Speech Signals*. Prentice-Hall signal processing series. Prentice-Hall, 1978. ISBN 9780132136037.

Xavier Serra and Julius O Smith III. Audio signal processing for music applications, 2014. URL <https://www.coursera.org/course/audio>.

Andreas Spanias, Ted Painter, and Venkatraman Atti. *Audio signal processing and coding*. John Wiley & Sons, 2006.

Albert Sweigart. *Invent Your Own Computer Games with Python: A Beginner's Guide to Computer Programming in Python*. Al Sweigart, 2010. ISBN 9780982106013. URL <http://inventwithpython.com/>.

Gael Varoquaux, Valentin Haenel, Emmanuelle Gouillart, Zbigniew Jędrzejewski-Szmek, Ralf Gommers, Fabian Pedregosa, Olav Vahtras, Pierre de Buyl, Gert-Ludwig Ingold, Nicolas P. Rougier, and et al. *scipy-lecture-notes*: Release 2015.1 beta, 2015. URL <http://www.scipy-lectures.org/>.

Момчило Богданов и Софија Богданова. *Дигитално процесирање на сигнали*. Електротехнички факултет, Скопје, 1997. ISBN 9989-630-15-1.