

Дигитални аудиосистеми

Бранислав Геразов

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

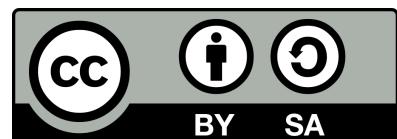
Дигитални аудиосистеми ~ Предавања и вежби v0.80

© Copyright by Бранислав Геразов 2016 – 2019 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 одбироци или семплирање	8
2.2 Дискретизација по амплитуда или квантизација	13
2.3 Формати на дигитално аудио	20
3 Вовед во процесирање на аудиосигналите	21
3.1 Вчитување на звук во Питон	21
3.2 Скратување на звукот и префрлање во моно	22
3.3 Прикажување на аудиосигналот	23
3.4 Преслушување на аудиосигналот	24
3.5 Менување на амплитудата на аудиосигналот	25
3.6 Нормализација на аудиосигналот	26
3.7 Генерирање на звук во Питон	27
4 Фреквенциски спектар на звучните сигнали	28
4.1 Основи на Фуриеовата анализа	28
4.2 Анализа на спектарот на звучните сигнали	35
4.3 Фуриеова трансформација на временски отсекоци ФТВО	36
5 Филтри	42
5.1 Основи на дигиталните филтри	42
5.2 Дизајн на ФИР филтер	46
5.3 Филтрирање на аудиосигнал со ФИР филтер	48
5.4 Дизајн на ИИР филтри	49
5.5 Употреба на ИИР филтри за еквализација	51
5.6 ИИР филтри непропусни на фреквенција – ноч филтри	53
5.7 Дигитални аудиоэффекти базирани на филтри	54
6 Процесирање на аудиосигналите базирано на доцнење	55
6.1 Основи на процесирање со задоцнување	55
6.2 Exo	56
6.3 Реверберација	58
6.4 Други дигитални ефекти базирани на доцнење	59
7 Компресија на аудиосигналите	60
7.1 MPEG-1 ниво III	61
7.2 Компресија на говор	64
8 Процесирање на аудиосигналите базирано на линеарна предикција	66
8.1 Анализа и синтеза на глас со линеарна предикција	67

Додаток А Слободен и отворен софтвер за инженерска и научна работа	72
A.1 Слободен софтвер	72
A.2 Четири слободи	72
A.3 Предности на слободниот софтвер	73
A.4 Одржливост	74
A.5 Слободен софтвер за инженерска и научна работа	75
Додаток А Питон за процесирање на аудиосигналите	76
A.1 Основи поставки во ГНУ/Линукс	76
A.2 Основи на работата со Питон	77
A.3 Основи на Нумпай и Матплотлиб	81

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 дистрибуција.

Дигиталните алатки се речиси семоќни кога станува збор за обработка на звучните записи. Од 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, може да кажеме дека опремата потребна за тоа е лесно достапна и масовно раширена, 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.

Поширок динамички опсег

Дигиталното аудио кодирано со 16–битна резолуција теориски нуди однос сигнал шум од 96 dB, споредено со динамичкиот опсег од околу 80 dB 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, зголемениот динамички опсег станува потреба, бидејќи менувањето на силината на влезниот сигнал за време на снимањето не е дозволено.

Зголемена отпорност на шум

Дигиталниот аудио запис со својата еднозначност овозможува отпорност 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.

Усовршено и олеснето умножување

Дигиталниот аудио запис, повторно поради својата еднозначност, може да се пресними од еден дигитален носач на звук на друг, без било каква загуба во квалитетот. За споредба, 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. Проблемот станува сериозен кога имаме синџир на преснимувања копија од копија од копија, при што квалитетот на конечната снимка ќе биде намален за сумата на штетни влијанија внесени од секој од уредите искористени во синџирот. Со други зборови квалитетот на аналогната снимка е условен од должината на синџирот, како и од квалитетот на уредите кои учествуваат во него. Дигиталното преснимување е отпорно на обата овие 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 и во најдобар случај мора да пројде онолку време колку што трае самиот звучен запис. Истото важи и кај преснимувањето од аудио касета на касета.¹ Што значи дека за преснимување 60-минутен запис скоро секогаш се потребни 60 минути за 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!²

Механизми за корекција на грешка

Погодно е преку примената 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.

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 максималните 30 години за магнетната лента. Кај златните и платинестите CD-а оваа граница оди и до 100, односно 200 години соодветно.

Неограничени можности за обработка и монтирање

Дигиталниот звукот за првпат 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

¹Некои системи нудат двократно скратување на ова време со зголемена брзина на движење на магнетните ленти при преснимувањето, но ова може да биде по цена на намалување на квалитетот на направената копија.

²Точните вредности зависат од параметрите на компјутерскиот хардвер. Тука се дадени заокружени вредности за да се даде перспектива на нивниот сооднос.

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 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 сеуште 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](#).

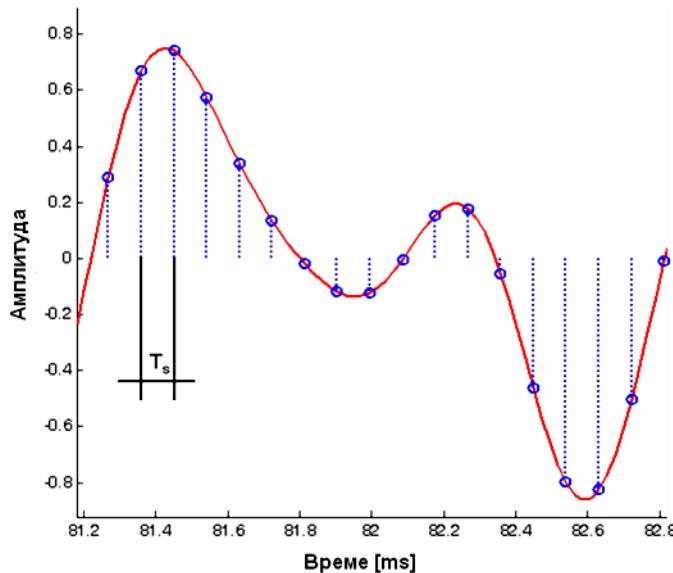
2.1 Земање одбироци или семплирање

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 за сигналот содржана во земените одбироци 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

¹ Важно 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 одбириоци со фреквенција f_s , односно периода T_s .

во време да се претстави со конечна низа на вредности? Што станува со информацијата за сигналот помеѓу одбириците? Сепак, математички е докажано дека ова е возможно. Станува збор за познатата **теорема за земање одбириоци**, која за првпат ја открива Владимир Котельников² и ја применува во телекомуникациите во 1933^{ta} година. До нејзе независно дошле и други научници, и тоа: Хари Нјуквист³, Клод Шенон⁴, и Едмунд Витакер⁵. Теоремата за земање одбириоци гарантира дека за веродостојно зачувување на целосната информација содржана во еден аналоген сигнал со максимална фреквенција на неговиот спектар f_m , доволно е од него да земаме одбириоци со фреквенција $f_s \geq 2f_m$. Минималната фреквенција на земање одбириоци $f_s = 2f_m$ се нарекува **Нјуквистова фреквенција**.

§ 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^{ta}, 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 $294 \times 50 \times 3 = 44.100$ примероци/секунда.

Спектарот на дискретизираниот сигнал е ист со овој на оригиналниот, со тоа што сега постојат копии на овој спектар центрирани на целобройни мултикли од фреквенцијата на земање на одбириоци 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.

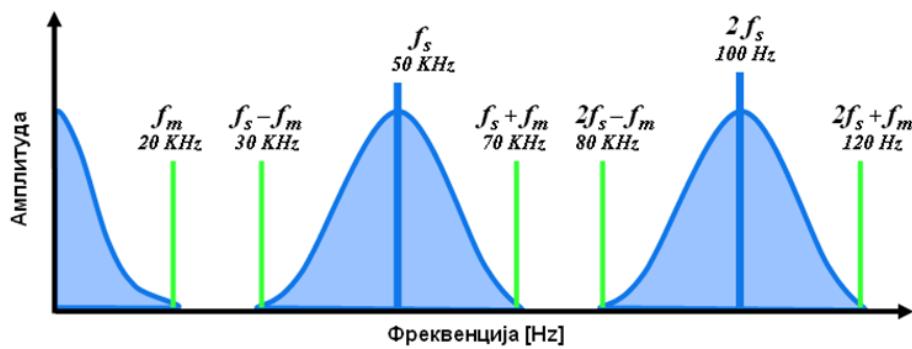
Бидејќи овие копии на спектарот на сигналот не смеат да се преклопуваат потребна е фреквенција на земање одбириоци поголема од $2f_m$, односно од Нјуквистовата фреквенција. Од

²Wikipedia: Vladimir Kotelnikov https://en.wikipedia.org/wiki/Vladimir_Kotelnikov

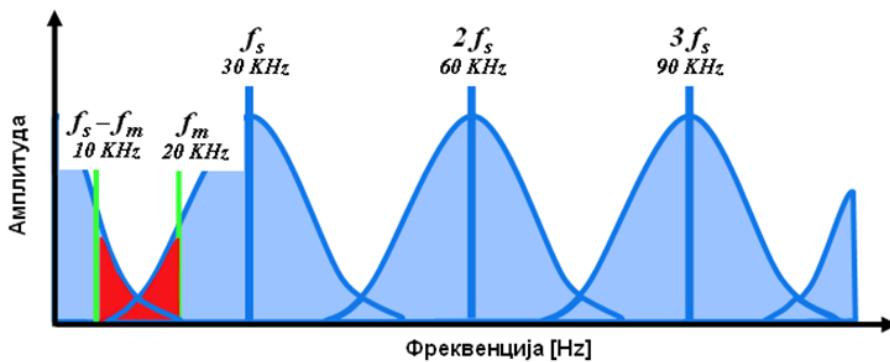
³Wikipedia: Harry Nyquist https://en.wikipedia.org/wiki/Harry_Nyquist

⁴Wikipedia: Claude Shannon https://en.wikipedia.org/wiki/Claude_Shannon

⁵Wikipedia: Edmund Taylor Whittaker https://en.wikipedia.org/wiki/E._T._Whittaker



Сл. 2.2: Спектар на дискретизираниот сигнал.



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

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 . 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, прикажано е на Сл. 2.3.⁶ 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.

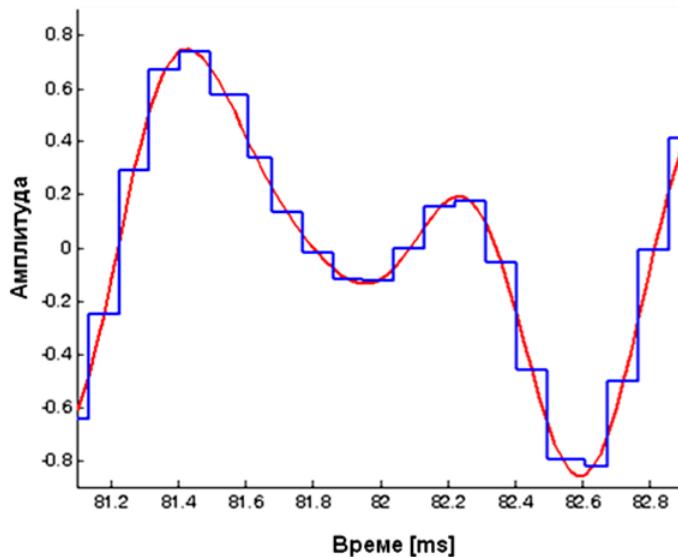
Реконструирањето на аналогниот сигнал од неговите одбирачи се врши со употреба на **ниско–пропусен (НФ) филтер** со гранична фреквенција f_m . Тој во фреквенциски домен има задача да ги отстрани сликите од спектарот на сигналот лоцирани околу мултиплите на фреквенцијата на земање одбирачи f_s , по што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

Фреквенции на семплирање во дигиталното аудио

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

- 8.000 Hz – се употребува во телефонската комуникација и е доволна за пренесување на говор со зачувана разбираливост,
- 11.025 Hz и 22.050 Hz – четвртина и половина од фреквенцијата на земање на одбирачи во аудио CD стандардот 44.100 Hz, се употребува за зачувување на дигитално аудио со понизок квалитет,

⁶Во англиската литература преклопувањето се нарекува **aliasing**, од зборот alias што значи лажно име, односно лажно претставување.



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

- 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 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 квантација. Овој начин на дигитизација не нуди предности над стандардниот кој е во општа примена.

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. Овој филтер треба 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 96 dB, всушност слабеењето на филтерот треба да е уште поголемо. Во практиката, ова подразбира употреба на аналоген филтер од 8th или 10th ред што е скапо и непрепорачливо решение, поради потребата за прецизна усогласеност на составните аналогни компоненти.

Постои уште еден поекономичен начин да се реализира земањето одбирачи, а тоа е со употребата на **надсемплирање**⁷. Надсемплирањето претставува земање на одбирачи со фреквенција неколку

⁷Во англиската терминологија тоа е познато како oversampling

пати поголема од Најквистовата. Со него се постигнува олеснување на условите кои влезниот НФ филтер 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 нископропусното 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 **ФИР**⁸ филтрите со конечен импулсен одсив. Потоа може да се отфрлат вишокот одбироци (одбироци), со што доаѓаме до целната фреквенција на земање одбироци.

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

Пример 1. За да ги илустрираме придобивките од надсемплирањето ќе претпоставиме $4\times$ надсемплирање во однос на аудио-CD стандардот, прикажано на Сл. 2.5. 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:

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

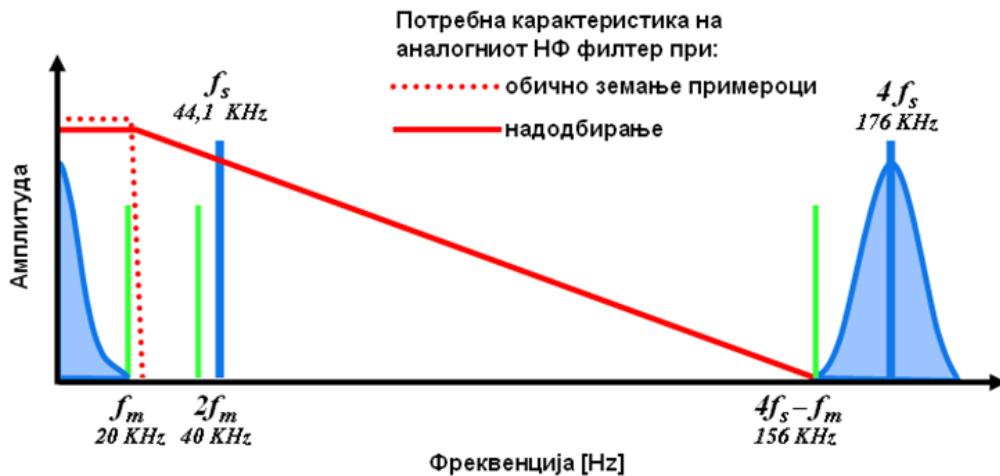
Така, аналогниот филтер може да биде со поблаг пад во карактеристиката и сепак да ги задоволи условите за висок квалитет на записот. Падот може да започне нешто над 20 kHz и да трае сè до 88^{ot} kHz, каде треба да го достигне слабеењето од 80 dB. Уште повеќе, нас не нè интересира верна репродукција 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. Следува дека сега преодниот фреквенциски опсег на филтерот изнесува 120 kHz споредено со 4,1 kHz во случајот кога не се користи надсемплирање. Ова во голема мера ја поедноставува изградбата на аналогниот филтер.

По надсемплирањето, сигналот ги следи останатите два чекора на дигитализацијата (квантизација и кодирање) со што од него се добива дигитален сигнал. При тоа, поради надсемплирањето, 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 треба да го сведеме дигиталниот сигнал на целната фреквенција на семплирање $f_s = 44,1$ kHz. Најпрвин за да го издвоиме корисниот спектар, дигиталниот сигнал треба да го исфилтрираме со дигитален НФ филтер со стрмно отсекување на фреквенциите над 22 kHz, кое 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 наречен **скастрување**⁹. 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

⁸Транслитерација од англиското FIR, што стои за Finite Impulse Response.

⁹Соодветниот английски термин е downsampling.



Сл. 2.5: Благиот наклон на влезниот НФ филтер овозможен со надодбирањето во споредба со потребната карактеристика на филтерот приично земање на одбирачи.

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. Кај DA конверторите надодбирањето се изведува со додавање на екстра нулти одбирачи на секој реален одбирок.^a При тоа нивната амплитуда најчесто се интерполира помеѓу вредностите на реалните одбирачи или пак математички се моделира, во зависност од изведбата на 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.

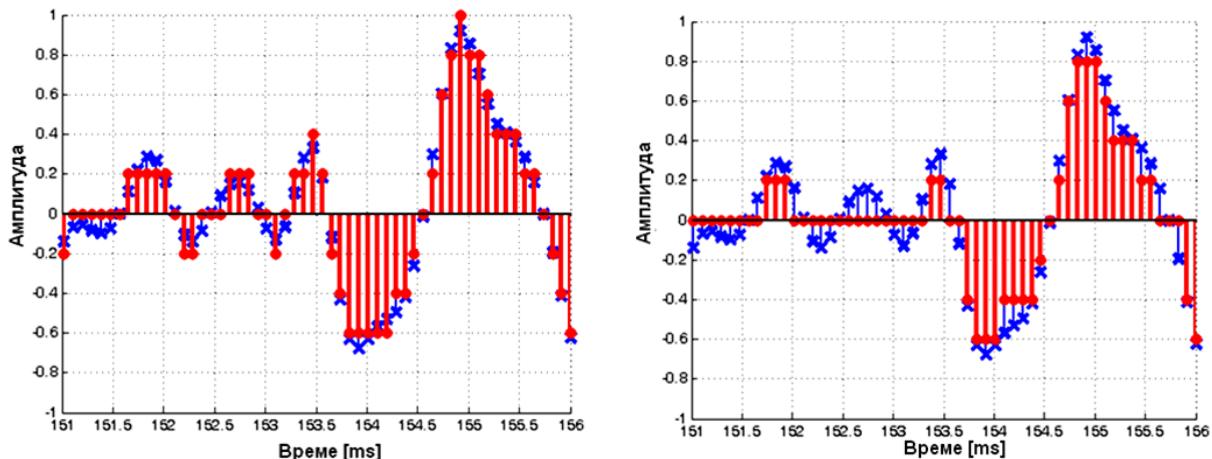
^aНа англиски ова се нарекува up-sampling.

Цитер

Цитерот претставува грешка во временските моменти во кои се прави семплирањето поради варијациите во временската база на уредот за дигитализација. Односно, AD конверторот со фреквенција на земање одбирачи од 44,1 kHz, не секогаш ги зема одбираците точно секој 44.100th 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 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 одбирачи 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, наречен **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 мапираат во ним најблиските им вредности во рамките на ова множество. Тој процес се нарекува **квантација**, а електронскиот



Сл. 2.6: Квантација (првено) на дискретизираниот сигнал (сино) со заокружување (лево) и со отсекување (десно).

уред со кој таа се изведува **квантизер**¹⁰. Бројот на битови кој квантизерот може да го генерира се нарекува **решолуција**.

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 – **MSB**¹¹, до најголемата прецизност која тој може да ја постигне, која е $V_{REF}/2^N$ и се кодира со битот со најмала тежина **LSB**¹². Процесот е воден од внатрешната логика на квантизерот. Важно е брзината со која се доаѓа до конечниот коден збор да е поголема од брзината со која доаѓаат одбираците 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.

Квантација на дигитално аудио

При дигитализација на аудиото се употребуваат следните решолуции 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 решолуција.
- **4 бита** – ретко се употребува при линеарна импулсно кодна модулација (ИКМ), а во употреба е кај адаптивната квантација.
- **8 бита** – во употреба во телефонските системи за кодирање на говорот при нелинеарна квантација, стандардна решолуција на старите звучни карти во ерата на флопи-

¹⁰ Влезниот аналоген филтер, земачот на примероци, квантизерот и на крај кодерот се составни делови на еден AD конвертор.

¹¹ Most Significant Bit.

¹² Least Significant Bit.

дискетите,¹³

- 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.

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. При тоа 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 **нелинеарната квантизација**, односно кодирање, кај која амплитудите со најголема веројатност на појавување пофино се квантизираат од оние кои поретко се појавуваат. На пример, за крајно високите амплитуди се отстапени помалку кодни зборови отколку за амплитудите поблиску до нулата.

Така, при квантизирање на говорот, посебно во класичните 8-битни телефонски системи, но и денес во врвните синтетизатори на говор како Вејвнет (Oord et al., 2016), вообичаено се употребува нелинеарна квантизација со крива за компресија, наречена и -крива.¹⁴ -кривата е дефинирана со G.711 стандардот на Меѓународната унија за телекомуникации и е описана со равенството:

$$f(x) = \text{sgn}(x) \begin{cases} \frac{A|x|}{1+\ln(A)}, & |x| < \frac{1}{A} \\ \frac{1+\ln(A|x|)}{1+\ln(A)}, & \frac{1}{A} \leq |x| \leq 1, \end{cases} \quad (2.4)$$

каде A во Европа е дефинирано да биде 87,6. На Сл. 2.7 е дадена преносната карактеристика на оваа крива. Како што може да се забележи, по компресијата поголем број на квантни нивоа се на располагање за квантација на помалите амплитуди на влезниот сигнал, а помал број за поголемите. Ова одговара на статистичката распределба на амплитудите на говорниот сигнал, во кој позастапени се малите амплитуди, па и од таму нужноста за нивна поточна квантација.

Исто така постојат и методи на квантација кои ја кодираат **разликата** помеѓу две последователни амплитуди на влезниот сигнал, така наречена адаптивна квантација. Тие се употребуваат во апликации каде големината на дигиталниот проток е критичен параметар.

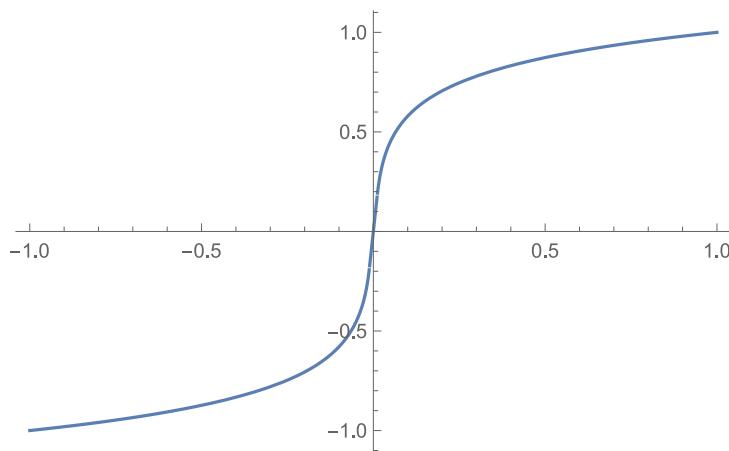
Шум на квантација

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 случаен процес и

¹³Денес се јавува во уметничкото движење кое употребува 8 битен звучен израз.

¹⁴A-law algorithm https://en.wikipedia.org/wiki/A-law_algorithm



Сл. 2.7: Графичка претстава на преносната карактеристика добиена со A кривата за нелинеарна квантација на говор. ¹⁵

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 Гаусов шум, односно тој ќе биде случаен и некорелиран со влезниот сигнал, со средна вредност 0 и максимална вредност $\pm q/2$ во случај на квантација со заокружување, односно средна вредност $q/2$ и максимална амплитуда q , 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 давава услови се исполнети речиси секогаш, особено за сигнали кои имаат мал коефициент на автокорелација, 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)¹⁶ на дигиталниот сигнал. Неговата вредност може да се добие со помош на равенството:

$$SNR = 10 \log \frac{P_x}{P_N} = 10 \log \frac{\sigma_x^2}{\sigma_q^2}, \quad (2.5)$$

каде P_x и P_N се моќноста на аудио сигналот и шумот на квантација, кои ако се случајни процеси можат да се пресметаат преку нивните стандардни девијации σ_x и σ_q .

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:

$$\sigma_q^2 = \frac{q^2}{12}. \quad (2.6)$$

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

$$\sigma_x^2 = \frac{1}{12 \cdot 2^{2N}}, \quad (2.7)$$

па (2.5) станува:

$$SNR = 10 \log \left(\frac{\sigma_x^2}{\frac{1}{12 \cdot 2^{2N}}} \right) = 10 \log(12 \cdot 2^{2N} \sigma_x^2). \quad (2.8)$$

¹⁵Преработено од [https://commons.wikimedia.org/wiki/File:Plot_of_F\(x\)_for_A-Law_for_A_%3D_87.6.svg](https://commons.wikimedia.org/wiki/File:Plot_of_F(x)_for_A-Law_for_A_%3D_87.6.svg) од Masterxilo1992 CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0/>

¹⁶Signal-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

При пресметување на равенството (2.8) 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 $\sigma_x = 0,25$ е:

$$SNR = 6,02n - 1,25 \simeq 6n \text{ [dB].} \quad (2.9)$$

Во праксата речиси секогаш се користи поедноставената 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 опсег на човекот.

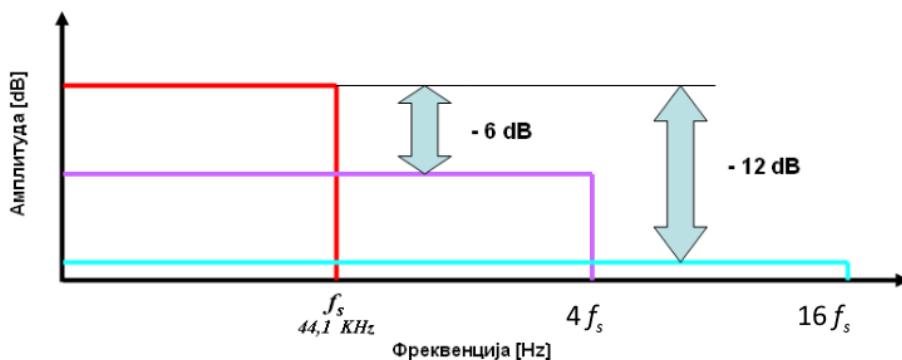
Квантизација при надсемплирање

Ако влезниот аналоген сигнал е случаен процес, па шумот на квантизација 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 квантизација 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. 2.8.¹⁷

Ова може да се каже и на следниов начин: 16-битен AD конвертор со брзина на земање одбироци од 44,1 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) стандардот¹⁸.

¹⁷Ова 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.

¹⁸SACD користи и техники за обликување на шумот дискутирани во Поглавјето за уште поголеми придобивки во SNR.



Сл. 2.8: Намалувањето на нивото на шумот со употреба на надсемплирање.

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

Дитер

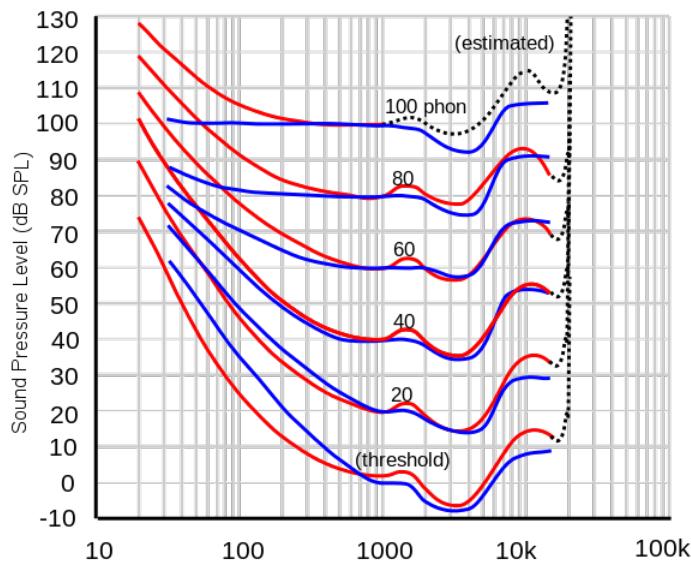
Претходната дискусија за нивото на шум на квантација важи во случај кога шумот на квантација е потполно некорелиран со влезниот сигнал. Во праксата, колку повеќе амплитудата на влезниот сигнал се приближува кон амплитудата означена со најмалку значајниот бит, толку повеќе се зголемува корелираноста на 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 па соодветно ќе внесе сопствени хармоници во спектарот на сигналот. Увото ги толкува овие изобличувања како непријатна дисторзија.

Дитерот¹⁹ е процес со кој може да се елиминира појавата на дисторзија во сигналот под дејство на шумот од квантација. Тој е воведен од Лоренс Робертс²⁰, 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 квантација 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

¹⁹Македонскиот 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).

²⁰Wikipedia: Lawrence Roberts https://en.wikipedia.org/wiki/Lawrence_Roberts_%28scientist%29



Сл. 2.9: Криви на еднаква чујност, или изофонски криви, измерени според Флечер и Мунсон (сино) и стандардизирани според ISO стандардот (црвено).²²

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 замасети и слепени. Подоцна минијатурни мотори – вибратори наречени dither-и се вградувале во сите механички компјутери.

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. 2.9. 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.²¹

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 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.

²¹Повеќе за нив може да чуете во делот **Психоакустика** во предметот **Електроакустика**.

²²"Lindos4" by Lindosland <https://commons.wikimedia.org/wiki/File:Lindos4.svg#/media/File:Lindos4.svg>

Пример 2. Каде $64\times$ надсемплирање во SACD стандардот, фреквенцијата на земање одбироци изнесува $2,8\text{ MHz}$, па шумот може да се распределат до $1,4\text{ MHz}$ што овозможува однос сигнал-шум од 100 dB при кодирање со 1 бит, што според (2.9) би требало да овозможи динамички опсег од само 6 dB .

2.3 Формати на дигитално аудио

По земањето на одбироци, квантацијата 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)**²³. 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 квантација 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 одбироци и резолуцијата на записот, додека кај аудио-CD-то овие информации се излишни. Форматот во кој е зачувано ИКМ аудиото во компјутерските системи зависи од оперативниот систем кој го користи компјутерот. Windows оперативните системи го користат **WAV**²⁴ форматот, развиен од страна на Микрософт за Интел базираните компјутерски системи во 1991, врз база на општиот RIFF (Resource Interchange File Format) дефиниран за чување на разни видови на податоци. Компјутерите со Macintosh платформи го употребуваат **AIFF**²⁵ 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 зачувуваат 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. Каде 16-битната квантација, 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.

²³Pulse Code Modulation (PCM).

²⁴Waveform audio format.

²⁵Audio Interchange File Format.

Поглавје 3

Вовед во процесирање на аудиосигналите

Со преминот на звукот во [дигитален домен](#), се отвораат вратите кон примена на различните техники на дигиталната обработка на сигналите за негова обработка.

Поради философијата на отвореност и соработка која е основа зад движењето за слободен софтвер, денес сè повеќе инженери и научници ја засноваат својата работа на платформи базирани на слободен софтвер. Повеќе за историјата, предностите и продорот на слободниот софтвер во инженерската и научно истражувачката практика може да прочитате во [Додатокот А](#).

Во духот на философијата зад слободниот софтвер, а следејќи ги светските инженерски и научни трендови, вежбите во предметот [Дигитални аудиосистеми](#) во целост ќе бидат изработени со слободен и отворен софтвер, поточно во слободниот програмскиот јазик [Питон](#). За вовед во екосистемот на Питон направен за научна работа и за основните системски поставувања погледнете во [Додатокот А](#).

Во оваа вежба ќе се запознаеме со основните принципи на работата со звукот во дигитален домен со употреба на програмскиот јазик Питон.

3.1 Вчитување на звук во Питон

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

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

Со овој код во матрицата `wav` сме ги вчитале одбираците на дигитализираниот звук сместени во звучниот фајл `Skopsko_stereo.wav`. Променливата во која ги вчитуваме звуците во Питон, 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 `wav` 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 `wav`.

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/local/lib/python3.6/site-packages/numpy/__init__.pyc'>
plt	module	<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/site-packages/matplotlib/pyplot.pyc'>
wav	ndarray	811782x2: 1623564 elems, type `int16`, 3247128 bytes (3 Mb)
wavfile	module	<module 'scipy.io.wavfile' from '/usr/local/lib/python3.6/site-packages/scipy/io/wavfile.pyc'>

Гледаме дека `wav` 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:

```
wav.shape[0]/fs
18.407755102040817
```

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

3.2 Скратување на звукот и префрлање во моно

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

```
wav = wav[: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:

```
wav = wav[:, 0]
```

или:

```
wav = wav[:, 1]
```

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

```
wav_mono = wav[:, 0] + wav[:, 1]
wav_mono = wav_mono / 2
```

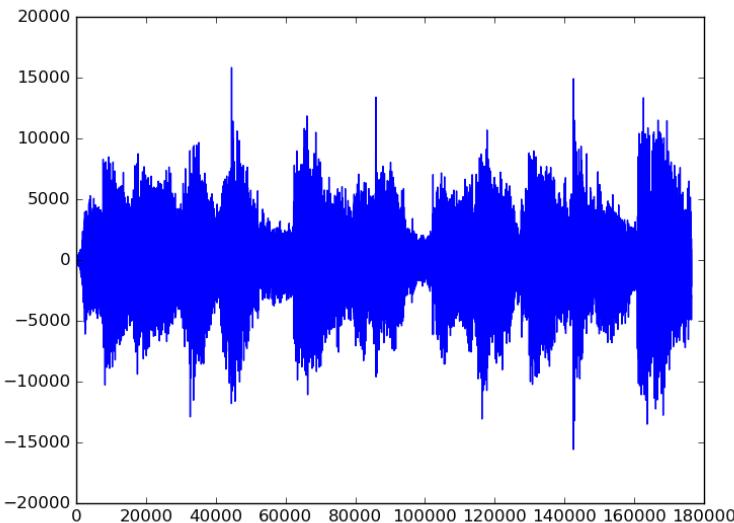
што може да се пресмета и како:

```
wav_mono = np.sum(wav, axis=1)
wav_mono = wav_mono / 2
```

односно наједноставно со:

```
wav_mono = np.mean(wav, axis=1)
```

Во функциите `np.sum` и `np.mean`, вториот параметар `axis` кажува по која оска да се изврши пресметката, со 1 е избрана првата димензија – односно по колони, т.е. по хоризонтала. Со 0 може да се избере нултата димензија - односно по редици, т.е. по вертикалата, која е и зададена автоматски.



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

3.3 Прикажување на аудиосигналот

За да го прикажеме аудиосигналот ќе го искористиме Матплотлиб:

```
plt.plot(wav_mono)
plt.show()
```

Со што ќе биде исцртан графиконот прикажан на Сл. 3.1. На овој графикон на x -оската е даден бројот на примерокот, а на y -оската неговата амплитуда. Бидејќи аудиосигналот 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 `wav_mono` со 2^{15} :

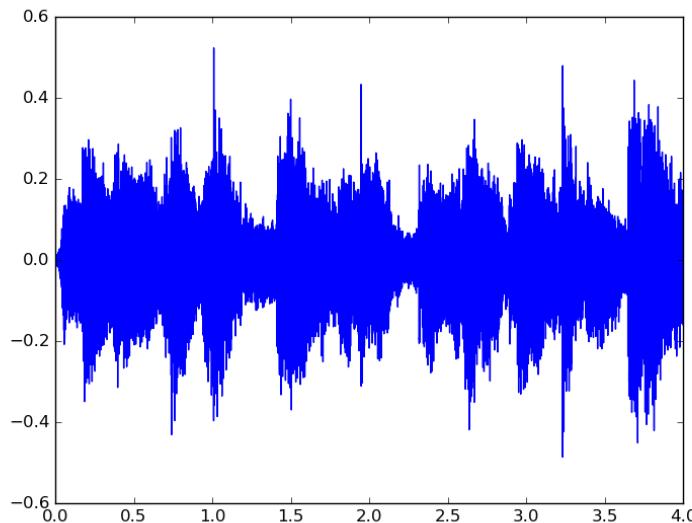
```
wav_mono /= 2**15
whos

Variable      Type      Data/Info
-----
...
wav_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.

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

Со што ќе биде исцртан графиконот прикажан на Сл. 3.2.



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

3.4 Преслушување на аудиосигналот

Наједноставниот начин да го преслушаме аудиосигналот е првии да го запишеме како аудио фајл, а потоа да ја искористиме системската `play` наредба од софтверскиот пакет Сокс.

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

Skopsko_mono.wav:

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

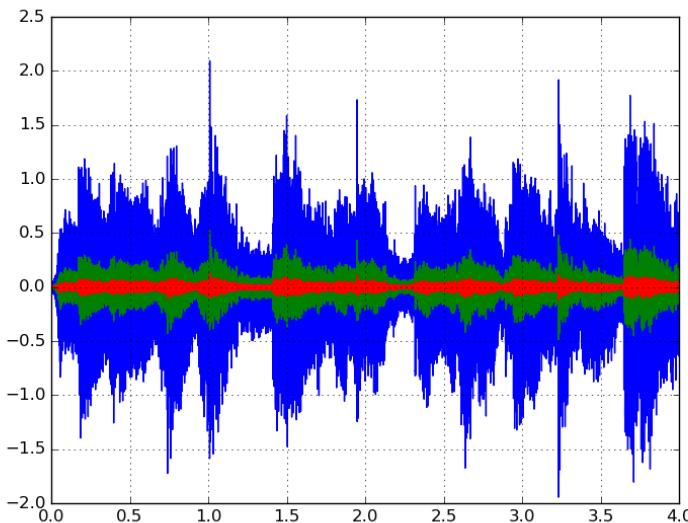
Може да забележиме дека Питон го запишал аудиофајлот со 64-битна прецизност наместо во оригиналната 16-битна. За ова да го поправиме треба да напишеме:

```
wavfile.write('Skopsko_mono.wav', fs, np.array(wav_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.
```



Сл. 3.3: Приказ на аудиосигналот `wav_mono` скалиран со различни коефициенти.

3.5 Менување на амплитудата на аудиосигналот

За понатамошна работа ќе ја искористиме Јупајтер Кјут конзолата, види [Додаток А](#). Во неа треба повторно да ги вчитаме потребните пакети:

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

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

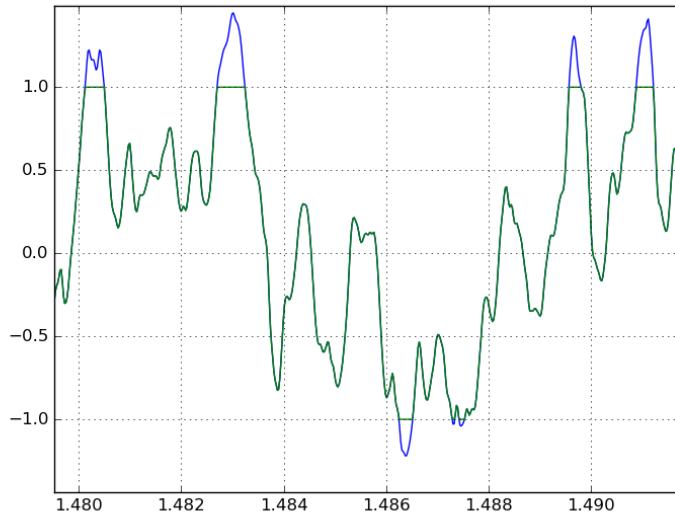
```
wav_glasno = wav_mono * 4
wav_tivko = wav_mono * .25
plt.figure()
plt.plot(t, wav_glasno)
plt.plot(t, wav_mono)
plt.plot(t, wav_tivko)
plt.grid()
```

Добиениот графикон е даден на Сл. 3.3. Може да видиме дека засилениот сигнал `wav_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.

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

```
wav_dist = wav_glasno.copy()
wav_dist[wav_dist > 1] = 1
```



Сл. 3.4: Приказ на дисторзија во аудиосигналот `wav_glasno`.

```
wav_dist[wav_dist < -1] = -1
plt.figure()
plt.plot(t, wav_glasno)
plt.plot(t, wav_dist)
plt.grid()
```

Копирањето во првата линија е нужно за да го задржиме оригиналниот вектор `wav_glasno`. Со приближување може да го добиеме приказот на Сл. 3.4.

3.6 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 од полна скала¹. Од (3.1) следува дека за максималната дозволена амплитуда аудиосигналот има ниво од 0 dBFS. Во праксата вообичаено звучните сигнали се нормализираат на амплитуда помала од 1, за при нивното понатамошно процесирање или пренос да не дојде до дисторзија.²

Да напишеме функција за нормализација:

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

која може да ја употребиме за нормализација на звучните сигнали:

¹На англиски *Full Scale*.

²Оваа амплитудна маргина која што се остава до дозволениот максимум на англиски се нарекува [headroom](#).

```
wav_mono_0dBFS = normalize(wav_mono, 0)
wav_mono_3dBFS = normalize(wav_mono, -3)
wav_mono.max()
wav_mono_0dBFS.max()
wav_mono_3dBFS.max()
```

3.7 Генерирање на звук во Питон

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

```
sound = np.sin(2*np.pi*200*t)
plt.plot(sound)
wavefile.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 притискање на `ctrl-shift-t` и стартувајте го стандардниот едитор `Плума` со наредбата:

```
$ pluma make_sound.py &
```

Во скриптата внесете го следниот код:

```
import numpy as np
from scipy.io import wavefile
import sys
import os

# Користејќи го модулот sys можеме на нашата скрипта да и пренесеме
# влезни параметри преку командната линија
print('sys.argv : ', sys.argv)
f = int(sys.argv[1])

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

и повикајте ја од ИПитон³:

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

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

³За 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 **Фурьеовата трансформација** и нејзината временски определена форма **Фурьеова трансформација на временски отсекочи (ФТВО)**¹. Ќе видиме како ФТВО на звучниот сигнал води до една прегледна форма за претставување на неговите временско/спектрални особини – **спектрограмот**.

4.1 Основи на Фурьеовата анализа

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



§ Дополнително. Францускиот математичар и физичар **Жан-Батис Жозеф Фурье**² (1768–1830) анализирајќи го ширењето на топлината, во 1822 за првпат постулирал дека која било функција може да биде претставена преку серија на синусоиди. Тој бил син на кројач и останал без својот татко на 9 годишна возраст. За време на француската револуција бил член на локалниот револуционерен комитет, за што и бил во затвор. Го придружувал Наполеон Бонапарта во неговиот поход во Египет во 1798, а по враќањето во Франција во 1801 станува претседател на општината Изер во Гренобл, каде започнува да прави експерименти за топлината.

¹Short-Time Fourier Transform (STFT).

²Wikipedia – Joseph Fourier https://en.wikipedia.org/wiki/Joseph_Fourier. Слика од Louis-Léopold Boilly - <https://www.gettyimages.com.au/license/169251384>, Public Domain, [https://commons.wikimedia.org/w/index.php?curid=3\char"0304\relax308441](https://commons.wikimedia.org/w/index.php?curid=3\char)

Табела 4.1: Позначајни својства на Фуриевата трансформација.

Својство	Временски домен	Фуриев домен
Линеарност	$ax(t) + bg(t)$	$aX(\omega) + bG(\omega)$
Поместување	$x(t - t_0)$	$e^{-j\omega_0 t} X(\omega)$
Модулација	$e^{-j\omega_0 t} x(t)$	$X(\omega - \omega_0)$
Конволуција	$x(t) * g(t) = \int_{-\infty}^{\infty} x(\tau)g(t - \tau) d\tau$	$X(\omega)G(\omega)$
Мултипликација	$x(n)g(n)$	$\frac{1}{2\pi} X(\omega) * G(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega)G(\omega - \Omega) d\Omega$

Фуриев интеграл

Фуриевата трансформација (ФТ) на сигналот $x(t)$ е дефинирана со интегралот:

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

каде со $\omega = 2\pi f$ е означена кружната фреквенција.

Инверзната Фуриева трансформација (ИФТ) е дефинирана со интегралот:

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

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

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

каде $A(\omega)$ претставува **амплитуден спектар**, а $\phi(\omega)$ **фазен спектар** на сигналот $x(t)$. 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:

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

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

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

Основните својства на Фуриевата трансформација се дадени во **Табелата 4.1**.

Сигналот $x(t)$ и неговата Фуриева трансформација $X(\omega)$ чинат **Фуриев пар** што се означува како $x(t) \leftrightarrow X(\omega)$. За Фуриевата трансформација се користи и симболот \mathcal{F} , односно пишуваме $\mathcal{F}\{x(t)\} = X(\omega)$, или за ИФТ $\mathcal{F}^{-1}\{X(\omega)\} = x(t)$.

Во **Табелата 4.2** се дадени некои позначајни Фуриеви парови.

Табела 4.2: Позначајни Фуриеови парови.

$x(t)$	$X(\omega)$
$\delta(t)$	1
1	$2\pi\delta(\omega)$
$\Pi(t) = \begin{cases} 1 & t < T \\ 0 & t > T \end{cases}$	$2T \frac{\sin(T\omega)}{T\omega}$
$\frac{\sin(\Omega t)}{\pi t}$	$\Pi(\omega) = \begin{cases} 1 & \omega < \Omega \\ 0 & \omega > \Omega \end{cases}$
$\cos(\omega_0 t)$	$\pi [\delta(\omega + \omega_0) + \delta(\omega - \omega_0)]$
$\sin(\omega_0 t)$	$j\pi [\delta(\omega + \omega_0) - \delta(\omega - \omega_0)]$

Енергијата на сигналот може да се пресмета преку неговиот спектар со употреба на теоремата на Парсевал³:

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(\omega)|^2 d\omega = \int_{-\infty}^{\infty} |X(2\pi ft)|^2 df \quad (4.7)$$

* Важно! Во Табелата 4.2 забележете дека Фуриевата трансформација на 1 е $2\pi\delta(\omega)$, додека на $\cos(\omega_0 t)$ е $\pi [\delta(\omega + \omega_0) + \delta(\omega - \omega_0)]$. Ова е во склад со теоремата на Парсевал дека енергијата во спектарот на косинусот помножена со коефициент $1/2\pi$ треба да е еднаква со онаа во временскиот сигнал. Поради тоа двата огледални Диракови импулси имаат двојно помала амплитуда, што не е случај кај константниот сигнал за кој имаме единствен Дираков импулс на фреквенција 0.

Фуриев ред

Ако функцијата $x(t)$ е периодична со периода T , односно основна фреквенција $f_0 = 1/T$, тогаш важи:

$$x(t) \equiv x(t + T), \quad \forall t. \quad (4.8)$$

Ако со $\hat{x}(t)$ го означиме сегментот на $x(t)$ во основниот интервал $t \in [-T/2, T/2]$:

$$\hat{x}(t) = x(t) \cdot \Pi(t), \quad (4.9)$$

каде:

$$\Pi(t) = \begin{cases} 1, & |t| \leq \frac{T}{2}, \\ 0, & |t| > \frac{T}{2}. \end{cases}, \quad (4.10)$$

тогаш можеме да ја изразиме $x(t)$ како сума од вакви функции $\hat{x}(t)$ поместени за мултили од T :

$$x(t) = \sum_{r=-\infty}^{\infty} \hat{x}(t - rT). \quad (4.11)$$

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

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk \frac{2\pi}{T} t} = \sum_{k=-\infty}^{\infty} c_k e^{jk \omega_0 t}, \quad (4.12)$$

³Wikipedia – Parseval's Theorem https://en.wikipedia.org/wiki/Parseval%27s_theorem

каде ω_0 е основната кружна фреквенција на сигналот, дадена со:

$$\omega_0 = 2\pi f_0 = \frac{2\pi}{T}. \quad (4.13)$$

Ова значи дека фреквенциите на комплексните синусоиди претставуваат мултикли од основната кружна фреквенција на сигналот ω_0 . Овие мултикли уште се нарекуваат и **хармоници** или **виши хармоници**, а f_0 и ω_0 се нарекува **основен хармоник**. Притоа кофициентите c_k кои претставуваат амплитуди на комплексните синусоиди можат да се добијат преку:

$$c_k = \frac{1}{T} \int_{-T/2}^{T/2} x(t) e^{-jk\frac{2\pi}{T}t} dt. \quad (4.14)$$

Фреквенцискиот спектар на ваков периодичен сигнал $x(t)$ претставува сума од поместени еквидистантни Диракови импулси $\delta(\omega)$:

$$X(\omega) = 2\pi \sum_{k=-\infty}^{\infty} c_k \delta(\omega - k\omega_0). \quad (4.15)$$

\mathcal{Z} трансформација

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

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}, \quad (4.16)$$

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

$$x(n) = x(nT_s), \quad (4.17)$$

$$T_s = \frac{1}{f_s}. \quad (4.18)$$

Тука, f_s е фреквенцијата на семплирање, а T_s е периодата на семплирање.

Инверзната \mathcal{Z} трансформација е дефинирана со:

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz, \quad (4.19)$$

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

Фуреова трансформација на дискретен сигнал

Фуреовата трансформација на дискретен сигнал може да се добие од равенката за \mathcal{Z} трансформацијата (4.16) со замената на јадрото $z = e^{-j\omega T_s}$, што се поедноставува на $z = e^{-j\omega}$ за $T_s = 1$:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n}, \quad (4.20)$$

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{j\omega n} d\omega. \quad (4.21)$$

Табела 4.3: Позначајни својства на \mathcal{Z} трансформација.

Својство	Временски домен	\mathcal{Z} 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	$x(n)\Pi(n)$	$\frac{1}{2\pi j} \oint_C X(\nu)W(z/\nu)\nu^{-1}d\nu$

Табела 4.4: Позначајни \mathcal{Z} парови.

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

Во овој случај, ја ограничуваме \mathcal{Z} трансформацијата на единечната кружница во z -рамнината опишана со $e^{-j\omega} = \cos(\omega) + j\sin(\omega)$, а дигиталната кружна фреквенција ω има интерпретација на агол во оваа рамнина. Со други зборови Фурьеовата трансформација на дискретниот сигнал претставува специјален случај на \mathcal{Z} трансформацијата.

Бидејќи новото јадро $e^{-j\omega}$ е периодична функција со периода 2π , следува дека фреквенцискиот спектар на дискретните сигнали е исто така периодичен со 2π во однос на кружната фреквенција ω . За произволно T_s , спектарот има периода $2\pi/T_s$ во однос на кружната фреквенција ω , односно периодата е $1/T_s = f_s$ во однос на фреквенцијата f , како што беше дискутирано во [Поглавјето 2.1](#).

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

Дискретна Фурьеова трансформација

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

$$\tilde{x}(n) = \tilde{x}(n + N), \quad -\infty < n < \infty, \quad (4.22)$$

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

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

$$\tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{jk\frac{2\pi}{N}n}. \quad (4.24)$$

Ако сега ја земеме Фурьеовата трансформација на дискретниот сигнал $x(n)$ дадена во [\(4.20\)](#), која претставува \mathcal{Z} трансформација на сигналот $x(n)$ долж единечната кружница, и истата ја семплираме на еквидистантни фреквенции ω_k дадени со:

$$\omega_k = k \frac{2\pi}{N}, \quad (4.25)$$

тогаш ќе го добијеме изразот за дискретната Фурьеова трансформација ДФТ:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-jk\frac{2\pi}{N}n}, \quad (4.26)$$

кој е еквивалентен со (4.23)!. Ова значи дека семплирирајќи спектар што го добиваме со (4.34) соодветствува на сигнал $\tilde{x}(n)$ која претставува периодична верзија од сигналот $x(n)$ со период N , односно важи:

$$x(n) = \tilde{x}(n) \cdot \Pi(n), \quad (4.27)$$

каде

$$\Pi(n) = \begin{cases} 1, & 0 \leq n < N, \\ 0, & \text{поинаку.} \end{cases} \quad (4.28)$$

Со други зборови, дискретниот спектар на еден дискретен сигнал $x(n)$ добиен со ДФТ претставува всушност спектар на периодичната верзија на сигналот во која тој се повторува бесконечно со периода еднаква на неговата должина N .

* Пример. Да земеме дека должината на дискретниот сигнал е $N = 4$, тогаш спектарот на сигналот добиен со ДФТ ќе биде пресметан за следните вредности на кружната фреквенција ω : $\omega_0 = 0$, $\omega_1 = \pi/2$, $\omega_2 = \pi$ и $\omega_3 = 3\pi/2$. Поради симетричноста на амплитудниот спектар за реални сигнали, што се сведува на симетричност на единечната кружница во однос на реалната оска во z рамнината, примероците ω_1 и ω_3 се огледални слики и се идентични. Поради тоа што тие не носат дополнителна информација, вообичаена практика е огледалните слики да се исфрлат, при што мора да се удвојат соодветните примероци кои остануваат за да се зачува валидноста на теоремата на Парсевал. По исфрлањето, последниот примерок од спектарот ќе биде за кружна фреквенција $\omega_3 = \pi$, односно, за произволно T_s , за фреквенција $f = f_s/2$. Така, конечната скратена верзија на ДФТ спектарот би се состоела од: $X(0)$, $2X(\omega_1)$ и $X(\pi)$.

За произволно T_s примероците ги земаме во кружните фреквенции:

$$\omega_k = k \frac{2\pi}{NT_s}, \quad (4.29)$$

односно:

$$f_k = k \frac{1}{NT_s} = k \frac{f_s}{N}. \quad (4.30)$$

Резолуцијата на спектарот добиен со ДФТ:

$$\Delta\omega = \frac{2\pi}{N}, \quad (4.31)$$

е одредена од должината на сигналот N . За произволно T_s таа ќе биде:

$$\Delta\omega = \frac{2\pi}{NT_s}, \quad (4.32)$$

односно:

$$\Delta f = \frac{f_s}{N}. \quad (4.33)$$

За зголемување на резолуцијата на спектарот добиен со ДФТ постојат две методи: *i*) да се земат повеќе примероци на сигналот N , и *ii*) да се додадат N_0 нули на крајот од сигналот. Последново ќе ја зголеми резолуцијата на семплирање во јадрото во (4.34), без да ги смени границите на сумата:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-jk\frac{2\pi}{N+N_0}n}. \quad (4.34)$$

Сепак, добиената резолуција нема да донесе нова информација во ДФТ спектарот, туку само ќе генерира интерполирана верзија на оригиналниот спектар со N точки.

Соодветно инверзната дискретна Фурьеова трансформација (ИДФТ) е дадена со изразот:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{jk \frac{2\pi}{N} n}. \quad (4.35)$$

За пресметка на ДФТ во праксата се користат алгоритми за нејзино брзо пресметување чија пресметковна сложеност е пропорционална со $N \log N$, кои се нарекуваат **брза Фурьеова трансформација (БФТ)** (Богданов, 1997).⁴ Бидејќи БФТ работи брзо само за должини на сигналот од $N = 2^M$, вообичаена пракса е додавањето нули на сигналот за да се задоволи овој услов.

Проценка на амплитудата на сигналот од нејзиниот спектар

За да видиме колкава е амплитудата на ФТ спектарот на еден дискретен сигнал ќе го пресметаме спектарот на една простопериодична низа $x(n)$ со кружна фреквенција ω_0 :

$$x(n) = A \cos(\omega_0 n) = A \cos(2\pi f_0 n). \quad (4.36)$$

Спектарот на овој сигнал е даден со равенството:

$$X(\omega) = \frac{A}{2} 2\pi (\delta(\omega + \omega_0) + \delta(\omega - \omega_0)). \quad (4.37)$$

Ако земеме отсечок на овој сигнал со должина од N примероци и ја пресметаме континуираната ФТ, тоа е еквивалентно на пресметка на ФТ врз производот на $x(n)$ со правоаголен прозорец $w(n) = \Pi(n)$ со траење $-N/2 \leq n < N/2$. Според својството на Фурьеовата трансформација според кое множењето на два сигнали во временски домен претставува нивна конволуција во Фурьеов домен, имаме:

$$\hat{X}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega) W(\omega - \Omega) d\Omega \quad (4.38)$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{A}{2} 2\pi (\delta(\Omega + \omega_0) + \delta(\Omega - \omega_0)) W(\omega - \Omega) d\Omega \quad (4.39)$$

$$= \frac{A}{2} (W(\omega + \omega_0) + W(\omega - \omega_0)), \quad (4.40)$$

каде $W(\omega)$ е ФТ на прозорецот дадена со:

$$W(\omega) = \frac{\sin(N \frac{\omega}{2})}{\sin(\frac{\omega}{2})}. \quad (4.41)$$

Според Лопиталовото правило⁵, кое вели дека во случаи кога за гранична вредност на функција се добива неопределено израз како $0/0$ или ∞/∞ , тогаш гранична вредност постои и таа се наоѓа преку изводите на двете функции, односно:

$$W(0) = \lim_{\omega \rightarrow 0} \frac{\sin(N \frac{\omega}{2})}{\sin \frac{\omega}{2}} = \lim_{\omega \rightarrow 0} \frac{\sin'(N \frac{\omega}{2})}{\sin' \frac{\omega}{2}} = \lim_{\omega \rightarrow 0} \frac{N \cos(N \frac{\omega}{2})}{\cos \frac{\omega}{2}} = N \quad (4.42)$$

Бидејќи амплитудата на спектарот на правоаголниот прозорец за $\omega \neq 0$ е мала во споредба со N за $\omega = 0$, спектарот на $\hat{x}(n)$ за фреквенција ω_0 ќе биде:

$$\hat{X}(\omega_0) = \frac{A}{2} (W(2\omega_0) + W(0)) \approx \frac{A}{2} N \quad (4.43)$$

⁴Wikipedia – Fast Fourier transform (FFT) https://en.wikipedia.org/wiki/Fast_Fourier_transform

⁵Википедија – Лопиталово правило https://mk.wikipedia.org/wiki/Лопиталово_правило

4.2 Анализа на спектарот на звучните сигнали

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.wav` кој е дел од аудиозаписите од предметот [Електроакустика](#)⁶, но го имаме и во проектниот фолдер за Дигитални аудиосистеми.

За нашата понатамошна работа ќе ги искористиме можностите на развојната средина Спајдер, види [Додаток А](#). За почеток, фајлот кој автоматски се отвора со стартирање на Спајдер 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.

```
"""
Дигитални аудио системи
Бежба 2 - спектар
Created on Wed Mar 23 23:21:08 2016
@author: das
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
import os

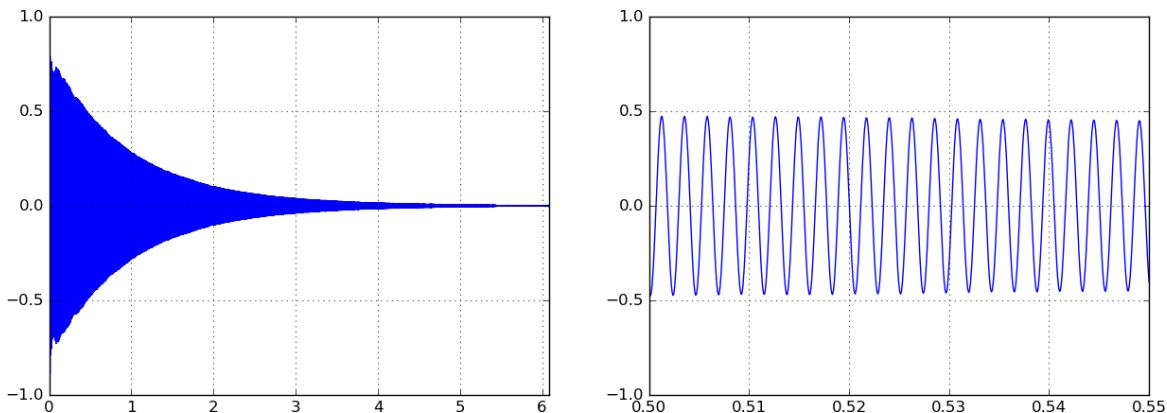
filename = 'audio/viluska.wav'
fs, wav = wavfile.read(filename)
os.system('play ' + filename)

wav = wav / 2**15
n = wav.size
t = np.arange(0, n)
t = t / fs
plt.figure(figsize=(15, 5))
plt.subplot(121) # 1X2 графици, прв график
plt.plot(t, wav)
plt.grid()
plt.axis([0, t[-1], -1, 1]) # [xmin, xmax, ymin, ymax]
plt.subplot(122) # 1X2 графици, втор график
plt.plot(t, wav)
plt.grid()
plt.axis([0.5, 0.55, -1, 1]) # [xmin, xmax, ymin, ymax]
```

Графиците кои ќе ги генерира дадениот код се прикажани на Сл. 4.1

* Важно! При првото извршување на Питон скрипта во едиторот на Спајдер 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.

⁶Материјалите за предметот Електроакустика на ФЕИТ може да ги најдете на <https://github.com/FEEIT-FreeCourseWare/Electroacoustics>



Сл. 4.1: Временски облик на аудиосигналот генериран од звучна вилушка.

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

За да го пресметаме спектарот на овој звучен сигнал ќе ја искористиме имплементацијата на БФТ алгоритамот во SciPy модулот `fftpack`, кој содржи и низа други функции за Фуриеова анализа. Во нашиот код ја додаваме секвенцата

4.3 Фуриеова трансформација на временски отсекоци ФТВО

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 ФТВО**.



§ Дополнително. Унгарскиот електроинженер **Денеш Габор**⁷ (1900–1979) е оној кој во 1946 ја адаптира Фуриеовата трансформација за временска анализа и ја добива ФТВО. 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.“

ФТВО 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 (4.44). 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

⁷Wikipedia: Dennis Gabor. https://en.wikipedia.org/wiki/Dennis_Gabor

прозорец по нивната должина се нарекува и метода на прозорци.

$$X_i(k) = \sum_{n=-N/2}^{N/2-1} w(n)x(n + i \cdot H)e^{-jk\frac{2\pi}{N}n}, \quad i = 0, 1, 2, \dots \quad (4.44)$$

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

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.⁸ 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 ФТВО анализата на аудиосигналите најупотребувани прозорци (Serra and O Smith III, 2014) се:

- Правоаголен прозорец

$$\text{rect}(n) = 1, \quad -N/2 \leq n \leq N/2 \quad (4.45)$$

$$\text{Rect}(k) = \text{sinc}(k) \quad (4.46)$$

- Хан прозорец

$$\text{hann}(n) = 0,5 + 0,5 \cos\left(2\pi \frac{n}{N}\right), \quad -N/2 \leq n \leq N/2 \quad (4.47)$$

$$\text{Hann}(k) = 0,5 \text{sinc}(k) + 0,25 (\text{sinc}(k-1) + \text{sinc}(k+1)) \quad (4.48)$$

- Хаминг прозорец

$$\text{hamm}(n) = 0,54 + 0,46 \cos\left(2\pi \frac{n}{N}\right), \quad -N/2 \leq n \leq N/2 \quad (4.49)$$

- Блекман прозорец

$$\text{black}(n) = 0,42 - 0,5 \cos\left(2\pi \frac{n}{N}\right) + 0,080,5 \cos\left(4\pi \frac{n}{N}\right), \quad -N/2 \leq n \leq N/2 \quad (4.50)$$

- Блекман-Харис прозорец

$$\text{blackharr}(n) = \frac{1}{N} \sum_{i=0}^3 a_i \cos\left(2i\pi \frac{n}{N}\right), \quad -N/2 \leq n \leq N/2 \quad (4.51)$$

каде

$$a_0 = 0,35876, a_1 = 0,48829, a_2 = 0,14128, a_3 = 0,01168 \quad (4.52)$$

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.

⁸Wikipedia: Windowing function https://en.wikipedia.org/wiki/Window_function

```

import numpy as np
from matplotlib import pyplot as plt
from scipy import fftpack as fft
from scipy import signal as sig

m = 512 # околина за анализа
n = 64 # должина на прозорец
m_half = m / 2
n_half = n / 2
w = np.zeros(m)
w[m_half-n_half : m_half+n_half] = sig.get_window('boxcar', n)
w_spec = fft.fft(w, m)
w_amp = np.abs(w_spec) / n
eps = 1e-9
w_amp[w_amp < eps] = eps
w_log = 20 * np.log10(w_amp)
w_log = w_log - np.max(w_log)
w_shift = fft.fftshift(w_log)

plt.figure(figsize=(12, 5))
plt.subplot(121)
plt.plot(w)
plt.axis([0, m, 0, 1]) # [xmin, xmax, ymin, ymax]
plt.grid()
plt.subplot(122)
plt.plot(w_shift)
plt.grid()
plt.axis([0, m, -100, 0])

```

Притоа за `eps` е земено произволно мала вредност со цел да ја избегнеме пресметката на логаритамот за вредности на спектарот близки до нулата. Вистинската нумеричка прецизност за `float64` може да се добие со Нумпай наредбата `np.finfo(float).eps` и таа изнесува $2,220446049250313 \cdot 10^{-16}$.

Различните прозорци и нивните амплитудни спектри генериирани со овој код се прикажани на Сл. 4.2. 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.

Спектrogram

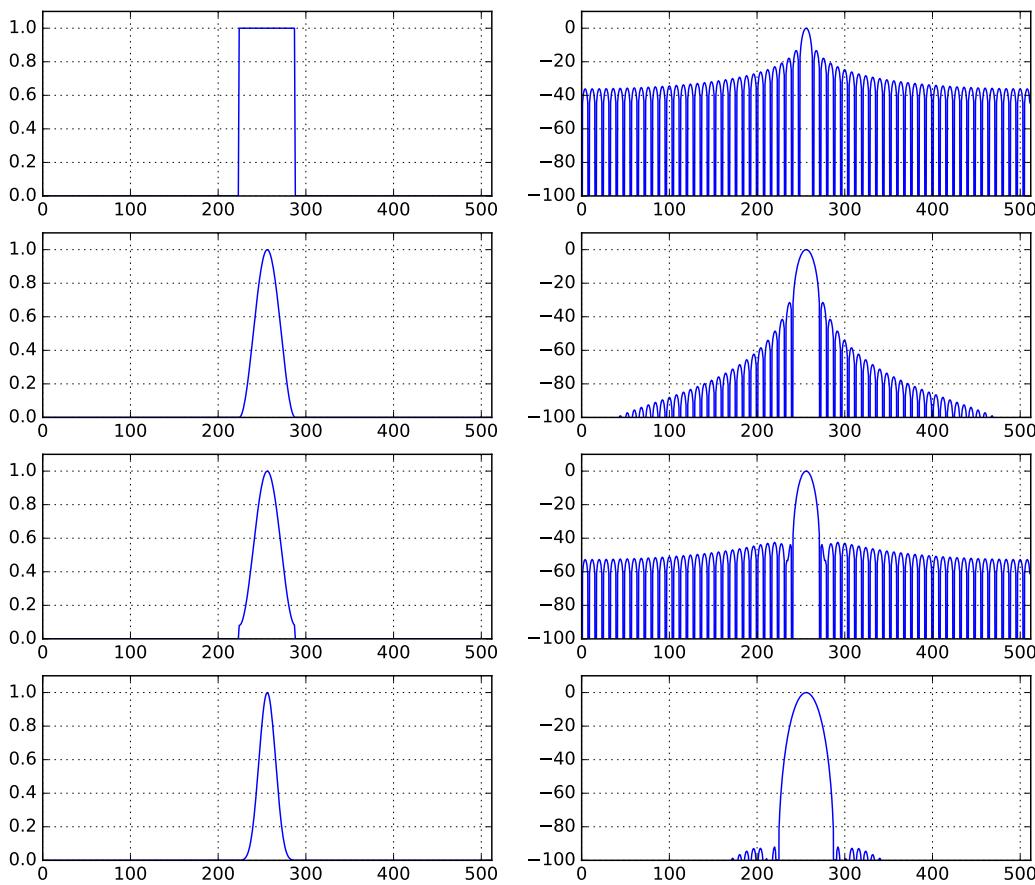
Спектrogramот е еден од најзначајните прикази на аудиосигналите. На него во исто време може да се набљудуваат карактеристиките на сигналот и во временски и во спектрален домен. За да го собиеме ќе се послужиме со ФТВО. Приказот е даден на Сл. 4.3.

```

import ...
import das

fs, wav = wavfile.read('audio/Zvona.wav')
wav = wav / 2**15
t = np.arange(0, wav.size/fs, 1/fs)
n = 2048 # 2**11
n_half = int(n/2)
h = int(n/2) # hop size
win = sig.get_window('hamming', n)
poz = n_half # позиција на средината на прозорецот

```



Сл. 4.2: Споредба на временскиот и спектрален облик на четири најчесто користени прозорци (од горе надолу): правоаголен, Ханов, Хамингов и Блекман-Харисов.

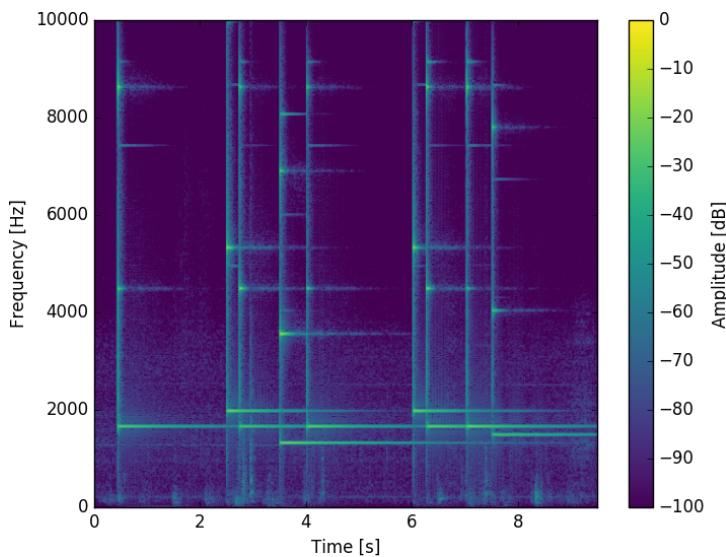
```

pad = np.zeros(n_half)
wav_pad = np.concatenate((pad, wav, pad))
m = wav_pad.size
frames_spec = None
while poz < m - n_half:
    frame = wav_pad[poz-n_half : poz+n_half] * win
    f_frame, frame_spec = das.get_spectrum(frame, fs)
    frame_spec_2d = frame_spec[:, np.newaxis]
    if frames_spec is None:
        frames_spec = frame_spec_2d
    else:
        frames_spec = np.hstack((frames_spec, frame_spec_2d))
    poz += h

plt.figure()
plt.imshow(frames_spec, aspect='auto',
           origin='lower',
           extent=[0, t[-1], 0, f_frame[-1]],
           vmin=-100, vmax=0, cmap='viridis')
plt.colorbar()
plt.xlabel('time [s]')
plt.ylabel('frequency [Hz]')
cbar.ax.set_ylabel('Amplitude [dB]')
plt.axis([0, t[-1], 0, 10000])

```

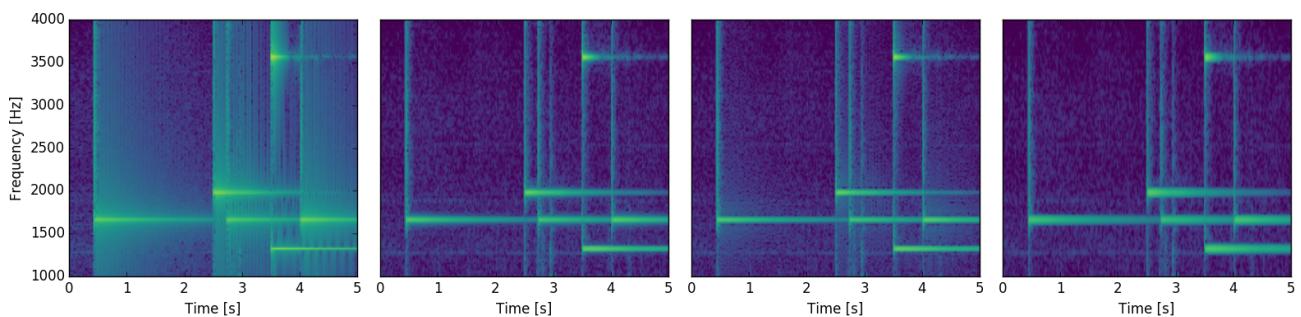
⁹Звукот Gentle Glockenspiel од bbatv е земен од Freesound.org. <http://freesound.org/people/bbatv/sounds/332932/>



Сл. 4.3: Спектрограм на аудиозаписот Zvona2.wav⁹ добиен со употреба на Хамингов прозорец со должина 2048 одбирачи (46 ms)

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

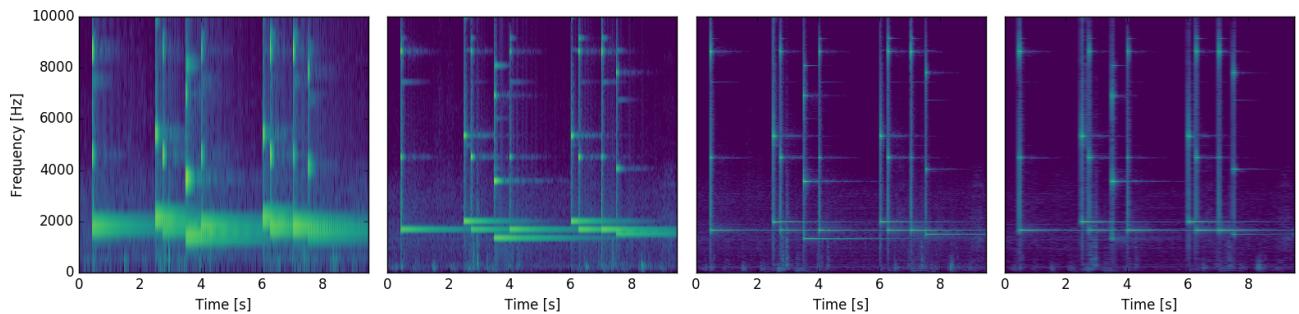
1. **Типот на прозорец.** Разликите во спектрограмот кои се добиваат со различните прозорци може да се воочат во деталите прикажани на Сл. 4.4.
2. **Големината на прозорецот.** Колку е поголем прозорецот толку повеќе точки, односно поголема резолуција, во спектарот на рамката земена од сигналот. Но исто така, колку поголем прозорец толку погруба временска резолуција на ФТВО анализата, Сл. 4.5. 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 ФТВО анализата.



Сл. 4.4: 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 мали 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 2015¹⁰.

¹⁰ Особено 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. 4.5: Спектрограми на аудиозаписот Zvona2.wav добиени со употреба на Хамингов прозорец со различни должини: 128, 512, 4096 и 8192 одбираци.

✓ Задача за час. Со помош на спектрограмот да се илустрира ефектот на преклопување на спектарот 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 АД конверзијата и за обликување на излезниот аналоген сигнал во ДА конверзијата; дигитални филтри имаме долж каналот за пренос на дигиталното аудио (трансмисија, дигитално снимање), а се основни градбени единки на еквализаторите. Аналогните филтри се реализираат со помош на збир на пасивни и активни електронски елементи, чија нагоденост е неопходна, но 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 системи¹. 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 (Богданов, 1997):

$$H\left\{\sum_{m=1}^M a_m x_m(n)\right\} = \sum_{m=1}^M a_m y_m(n), \quad \forall a_m \text{ -- линеарност и} \quad (5.1)$$

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

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 фреквенциска преносна функција во Фуриев 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:

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

Типови филтри според должината на импулсниот одсив

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

- конечен импулсен одсив (**ФИР**²) и
- бесконечен импулсен одсив (**ИИР**³).

ФИР филтрите имаат одредени предности над ИИР филтрите, а тоа се пред сè нивната стабилност и нивната линеарна фазна карактеристика. ИИР 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**. ИИР филтрите секогаш се реализираат со повратна врска, од каде произлегуваат и проблемите со нивната стабилност. ФИР 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 праксата овој услов секогаш се обезбедува преку нормализирање на останатите a коефициенти. Бројот на коефициенти во повратната врска на ИИР 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:

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

Типови филтри според амплитудната карактеристика

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

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

²Од англиското Finite impulse response.

³Од англиското Infinite 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 (НП),
- високопропусни (ВП),
- пропусни на опсег (ПО),
- непропусни на опсег (НО) и
- сепропусни (СП).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.

Типови ФИР филтри со линеарна фазна карактеристика

Строго линеарната фазна карактеристика не може да биде постигнато со ИИР или со аналогните филтри (Богданов, 1997), туку само со ФИР филтри чив 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 ИИР филтрите.

Постојат четири типови на ФИР 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:

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

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 ФИР 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:

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

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 високопропусен или непропусник 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:

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

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:

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

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 или пропусник на опсег.

Постојат различни пристапи за [дизајн на дигитални филтри](#). Трите најпознати методи за дизајн на ФИР филтри се:

- метода на прозорци,

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

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

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

5.2 Дизајн на ФИР филтер

Дизајнот на ФИР 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:

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

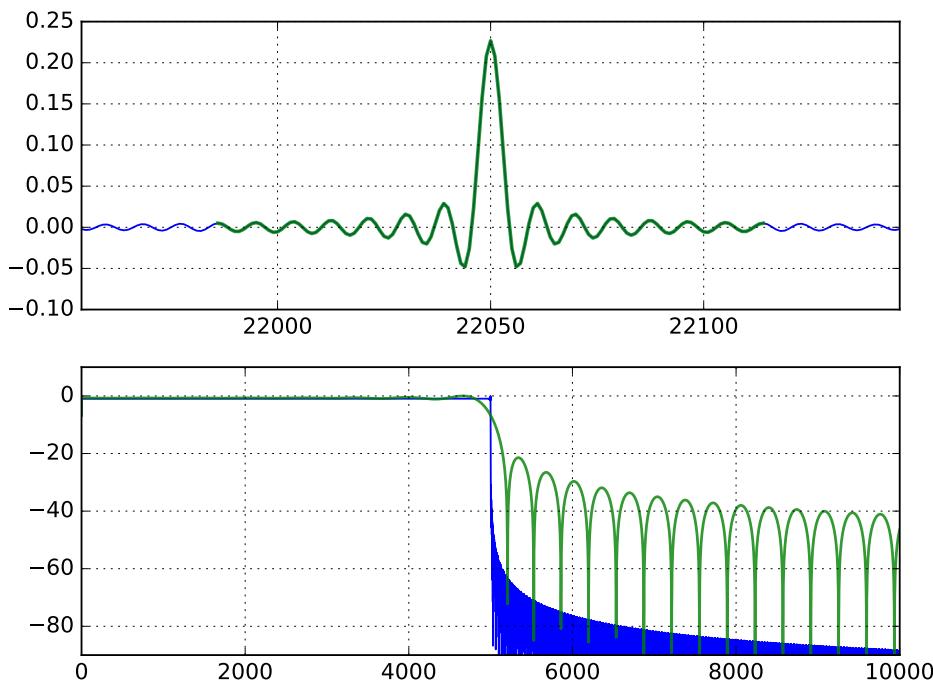
Овој импулсен одсив не може практично да се реализира поради тоа што неговото траење е бесконечно и тој не е каузален. За таа цел во методата на прозорци се зема само дел од него преку негово множење со избран прозорец. И тука важат истите дискусиии во [Поглавјата](#) и за влијанието на спектралните 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 `n_fft = fs` точки, неговиот импулсен одсив ќе го пресмета со употреба на ИДФТ, 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 ФИР филтер всушност претставува комбинација од методите за дизајн базирани на ДФТ и примената на прозорци.⁵

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import fftpack as fft
from scipy import signal as sig

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

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



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

```
# %% метода на прозорци
n = 128 + 1 # должна на прозорецот = ред на филтерот
nh = (n-1) / 2
n_ffth = n_fft/2

win = sig.get_window('boxcar', n) # правоаголен прозорец
n_win = np.arange(n_ffth - nh, n_ffth + 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, lw=2, alpha=.8)
plt.grid()
plt.subplot(212)
f, h_spec_l = das.get_spectrum(h_l, fs)
plt.plot(f, h_spec_l)
f, h_spec_rect = das.get_spectrum(h_rect_long, fs)
h_spec_rect = h_spec_rect - np.max(h_spec_rect) # нормализација
plt.plot(f, h_spec_rect, lw=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.

§ 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 конструираме има `fs`. Всушност и тој самиот како да сме го добиле со методата на прозорци од вистинскиот.

5.3 Филтрирање на аудиосигнал со ФИР филтер

Во овој дел ќе исфилтрираме еден звучен сигнал со ФИР филтер креиран со методата на прозорци имплементирана во функцијата `scipy.signal.firwin`.

```
import numpy as np
from scipy import signal as sig
from scipy.io import wavfile
import matplotlib.pyplot as plt
import os
import das

# %% вчитај го аудиосигналот
audio_path = '../audio/'
file_name = 'Mara.wav'
fs, wav_orig = wavfile.read(audio_path + file_name)
os.system('play ' + audio_path + file_name)
wav_orig = wav_orig / 2**15

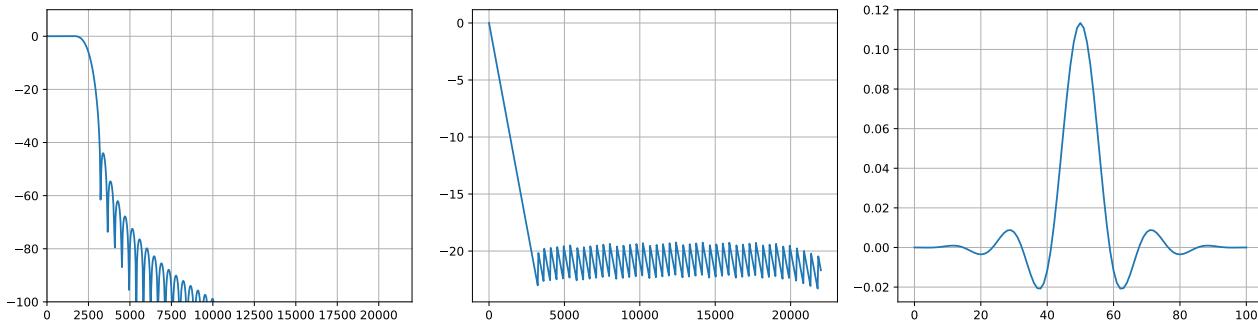
# %% исцртај спектрограм
__ = das.get_spectrogram(fs, wav_orig)

# %% дизајнирај ФИР филтер
order = 100
f_l = 2500
b = sig.firwin(order+1, f_l / (fs/2), pass_zero=True, window='hann')

# %% исцртај ја неговата фреквенциска карактеристика
w, h_w = sig.freqz(b)
f = w / np.pi * fs/2
h_amp = 20 * np.log10(np.abs(h_w))
h_ph = np.unwrap(np.angle(h_w))

plt.figure(figsize=(15, 4))
plt.subplot(131)
plt.plot(f, h_amp)
plt.grid()
plt.axis([0, f[-1], -100, 10])
plt.subplot(132)
plt.plot(f, h_ph)
plt.grid()
plt.subplot(133)
plt.plot(b)
plt.grid()
plt.tight_layout()
```

Карактеристиките на дизајнираниот ФИР филтер со ред $N = 100$ се прикажани на Сл. 5.2. Според неговата амплитудната карактеристика, може да видиме дека филтерот е нископропусен со гранична фреквенција f_l од 2,5 kHz. Исто така може да го видиме влијанието на Хановиот прозорец врз амплитудната карактеристика. Исто така може да видиме дека тој има строго линеарна фазна карактеристика, односно внесува подеднакво доцнење на компонентите на



Сл. 5.2: Амплитудна (лево) и фазна карактеристика (средина) и импулсен одсив (десно) на дизајнираниот ФИР филтер од 100-ти ред со употреба на Ханов прозорец.

сигналот долж сите фреквенции. Конечно можеме да го видиме неговиот одсив којшто поради каузалноста почнува од нулата со врвна амплитуда за $n = 51$, поради што филтерот внесува поместување, односно доцнење во излезниот сигнал.

✓ **Задача за час.** Проверете како се менуваат карактеристиките на филтерот со промена на:

- границната фреквенција,
- редот на филтерот, и
- типот на прозорецот.

Ајде сега да го исфилтрираме нашиот сигнал и да видиме како ќе се промени неговата спектротемпорална содржина.

```
# %% исфилтрирај го аудиосигналот и прикажи го спектрограмот на излезниот сигнал
wav_filt = sig.lfilter(b, 1, wav_orig)
__ = das.get_spectrogram(fs, wav_filt)

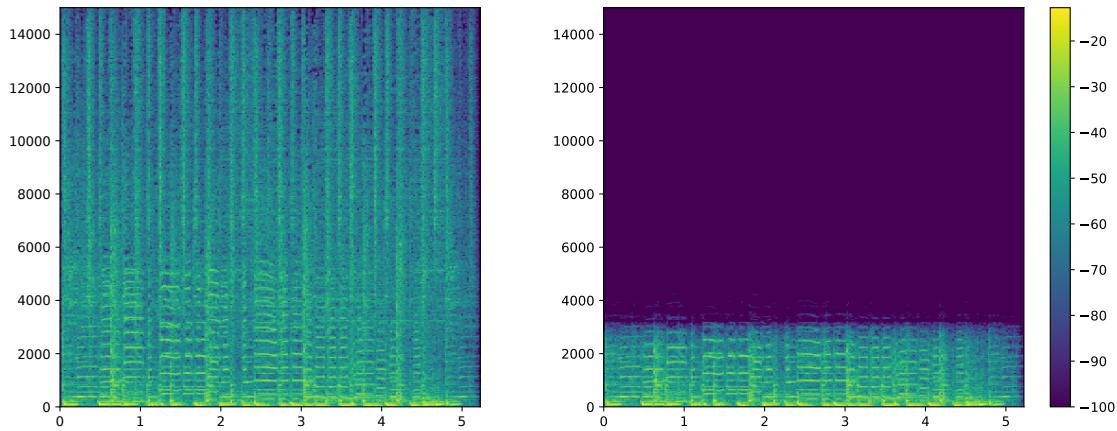
# %% сними и преслушај го добиениот аудиосигнал
wav_16 = wav_filt * 2**15
wav_16 = wav_16.astype('int16')
wavfile.write(audio_path + 'mara_fir.wav', fs, wav_16)
os.system('play ' + audio_path + 'mara_fir.wav')
```

Спектрограмот на вчитаниот и исфилтрираниот аудиосигнал се прикажани на Сл. 5.3. Може да видиме дека се работи за сложен аудиосигнал во кој се измешани периодични и апериодични звучни сигнали. Притоа, бидејќи спектрограмот не ни требаше понатаму во нашиот код, излезот на функцијата `das.get_spectrogram` го доделивме на променливата со име `__` што вообичаено се прави во вакви случаји. Во исфилтрираниот аудиосигнал може да забележиме дека филтерот ефикасно ги отстранил сите фреквенциски компоненти на сигналот над 2,5 kHz, како што може и да чуеме во излезниот аудиосигнал.

5.4 Дизајн на ИИР филтри

За дизајн на ИИР филтри ќе се послужиме со функцијата `scipy.signal.iirfilter` која содржи имплементации на дизајнот на ИИР филтри според споменатите методите на Батерворт, Бесел, Чебишев, и онаа на елиптичните филтри. Ќе ја искористиме методата на Батерворт која дава филтри со строго монотона фреквенциска карактеристика.

```
# %% дизајн на ИИР филтер
order = 10
f_1 = 2500
```



Сл. 5.3: Спектрограм на аудиосигналот `Mara` пред (лево) и по филтрирањето со ФИР филтерот (десно).

```
b, a = sig.iirfilter(order, f_l / (fs/2), btype='lowpass', ftype='butter')
w, h_w = sig.freqz(b, a)
f = w/np.pi * fs/2
h_amp = 20 * np.log10(np.abs(h_w))
h_ph = np.unwrap(np.angle(h_w))

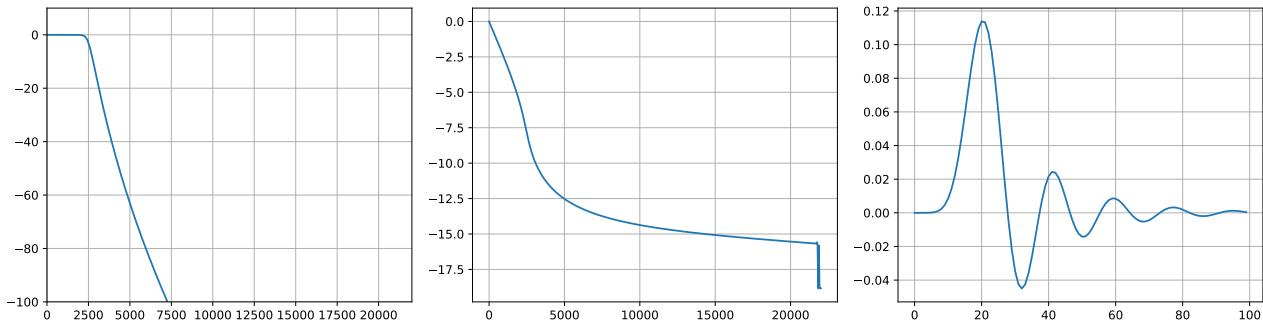
excite = np.zeros(100)
excite[0] = 1
h_n = sig.lfilter(b, a, excite)

plt.figure(figsize=(15, 4))
plt.subplot(131)
plt.plot(f, h_amp)
plt.grid()
plt.axis([0, f[-1], -100, 10])
plt.subplot(132)
plt.plot(f, h_ph)
plt.grid()
plt.subplot(133)
plt.plot(h_n)
plt.grid()
plt.tight_layout()
```

Забележете дека функцијата `iirfilter` ги враќа `b` но и `a` коефициентите на дизајнираниот ИИР филтер, двата $N + 1$ на број. Исто така, бидејќи импулсниот одсив не може сега да се добие директно од коефициентите на филтерот, првин создаваме побуден сигнал во форма на Дираков импулс `excite` кој потоа го филтрираме со коефициентите на филтерот за да го добиеме неговтиот импулсен одсив `h_n`.

На Сл. 5.4 се прикажани карактеристиките на дизајнираниот ИИР филтер со ред $N = 10$. Споредено со амплитудната карактеристика на ФИР филтерот од 100-ти ред, може да видиме дека ИИР филтерот има споредливи перформанси со 10 пати помал ред! Поради ова ИИР филтрите почесто се користат во процесирањето на аудиосигналите, и покрај нелинеарната фазна карактеристика, којашто може да се види на сликата. Како додатна предност, Батерворт филтрите ги немаат бранувањата во амплитудната карактеристика кои ги внесува прозорецот при дизајнот на ФИР филтрите.

Конечно, можеме да видиме дека импулсниот одсив на дизајнираниот ИИР филтер не е бесконечен, односно за одредено време неговата амплитуда може да се занемари. Всушност времетраењето на импулсниот одсив на ИИР филтеров е споредливо со она на претходно дизајнираниот ФИР филтер.



Сл. 5.4: Амплитудна (лево) и фазна карактеристика (средина) и импулсен одсив (десно) на дизајнираниот ИИР Батерворт филтер од 10-ти ред.

✓ **Задача за час.** Проучете ја промената на карактеристиките на ИИР филтерот за различни редови на филтерот. Да ли може да се дизајнира стабилен ИИР филтер од произволно голем ред?

5.5 Употреба на ИИР филтри за 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 помала 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 подржуваат 6 и повеќе-канална еквализација. Притоа дигиталните филтри се најчесто од ИИР тип од 4 ред, за намалување на комплексноста. Притоа, треба да се обрне особено внимание на нелинеарната фазна карактеристика на ИИР филтрите.

Еквализаторот во Питон 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.

```

# %% дефинирање на филтрите
n = 7 # ред на филтрите
# бас
f_b = 400 / (fs/2)
b_b, a_b = sig.iirfilter(n, f_b, btype='low', ftype='butter')
# средни
f_ml = 400 / (fs/2)
f_mh = 4000 / (fs/2)
b_m, a_m = sig.iirfilter(n, [f_ml, f_mh], btype='band', ftype='butter')
# треба
f_h = 4000 / (fs/2)
b_t, a_t = sig.iirfilter(n, f_h, btype='high', ftype='butter')

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

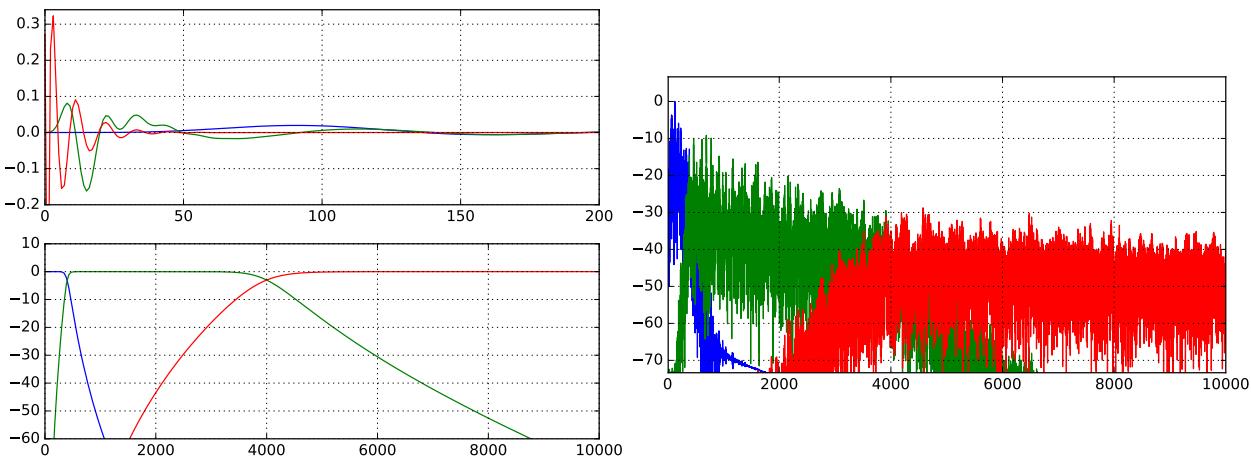
# %% импулсни одсиви
x = np.zeros(1000)
x[0] = 1
h_b = sig.lfilter(b_b, a_b, x)
h_m = sig.lfilter(b_m, a_m, x)
h_t = sig.lfilter(b_t, a_t, x)

# %% плотирање
plt.figure()
plt.subplot(211)
plt.plot(h_b)
plt.plot(h_m)
plt.plot(h_t)
plt.axis([0, 200, -.2, .34])
plt.grid()
plt.subplot(212)
plt.plot(f, H_b)
plt.plot(f, H_m)
plt.plot(f, H_t)
plt.axis([0, 10000, -60, 10])
plt.grid()

# %% филтрирање
fs, wav = wavfile.read('audio/Mara.wav')
wav_bass = sig.lfilter(b_b, a_b, wav)
wav_mid = sig.lfilter(b_m, a_m, wav)
wav_treble = sig.lfilter(b_t, a_t, wav)

# %% засилување
g_bass = -20 # во dB
g_mid = 0 # во dB
g_treble = 20 # во dB
g_b = 10** (g_bass/20)
g_m = 10** (g_mid/20)

```



Сл. 5.5: Импулсни одсиви и фреквенциски карактеристики на трите ИИР филтри од дизајнираниот еквализатор и добиени подопсези од аудиосигналот.

```

g_t = 10** (g_treble/20)
wav_out = g_b*wav_bass + g_m*wav_mid + g_t*wav_treble

# %% плотирање
f, wav_spec = das.get_spectrum(wav, fs)
f, wav_bass_spec = das.get_spectrum(wav_bass, fs)
f, wav_mid_spec = das.get_spectrum(wav_mid, fs)
f, wav_treble_spec = das.get_spectrum(wav_treble, fs)
plt.figure()
plt.plot(f, wav_spec)
plt.plot(f, wav_bass_spec)
plt.plot(f, wav_mid_spec)
plt.plot(f, wav_treble_spec)
plt.grid()

# %% преслушување
import os
wav_out = wav_out / np.max(np.abs(wav_out))
wav_out = wav_out * 2**15
wavfile.write('audio/Mara.eql.wav', fs, wav_out.astype('int16'))
os.system('play audio/Mara.wav')
os.system('play audio/Mara.eql.wav')

```

Импулсните одсиви, фреквенциските карактеристики на трите дизајнирани филтри и добиените подопсези од аудиосигналот се прикажани на Сл. 5.5.

5.6 ИИР филтри непропусни на фреквенција – ноќ филтри

Една од примените на дигиталните филтри за која неприкосновени се ИИР филтрите е потиснувањето на одредена фреквенција во аудиосигналите. Најчесто потребата за ваков тип на процесирање се јавува кога треба да се потисне хармоничниот шум од градската мрежа на 50 Hz, познат како **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 слабеење, односно голем **Q фактор**. Овие типови на филтри се нарекуваат **ноќ филтри**⁶.

⁶На англиски *notch* филтер

5.7 Дигитални аудиоекти базирани на филтри

Најпознатите дигитални аудиоекти базирани на филтри се **ва-ва** и **фејзерот**. Ва-ва ефектот⁷ 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).

Фејзерот⁸ 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 слабеење 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 дискутиирани во поглавјето 6.1.

⁷Wikipedia: Wah-wah (music). https://en.wikipedia.org/wiki/Wah-wah_%28music%29

⁸Wikipedia: 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 [ревербацијата](#). 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 задоцнета верзија на аудиосигналот и истата ја додава на него може да го представиме 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 задоцнетата 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 неполовна 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 мултипли од доцнењето $m \times D$ на првото доцнење. 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 генерираше 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 exo**. 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 exo**.

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

```

Сега 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)

```

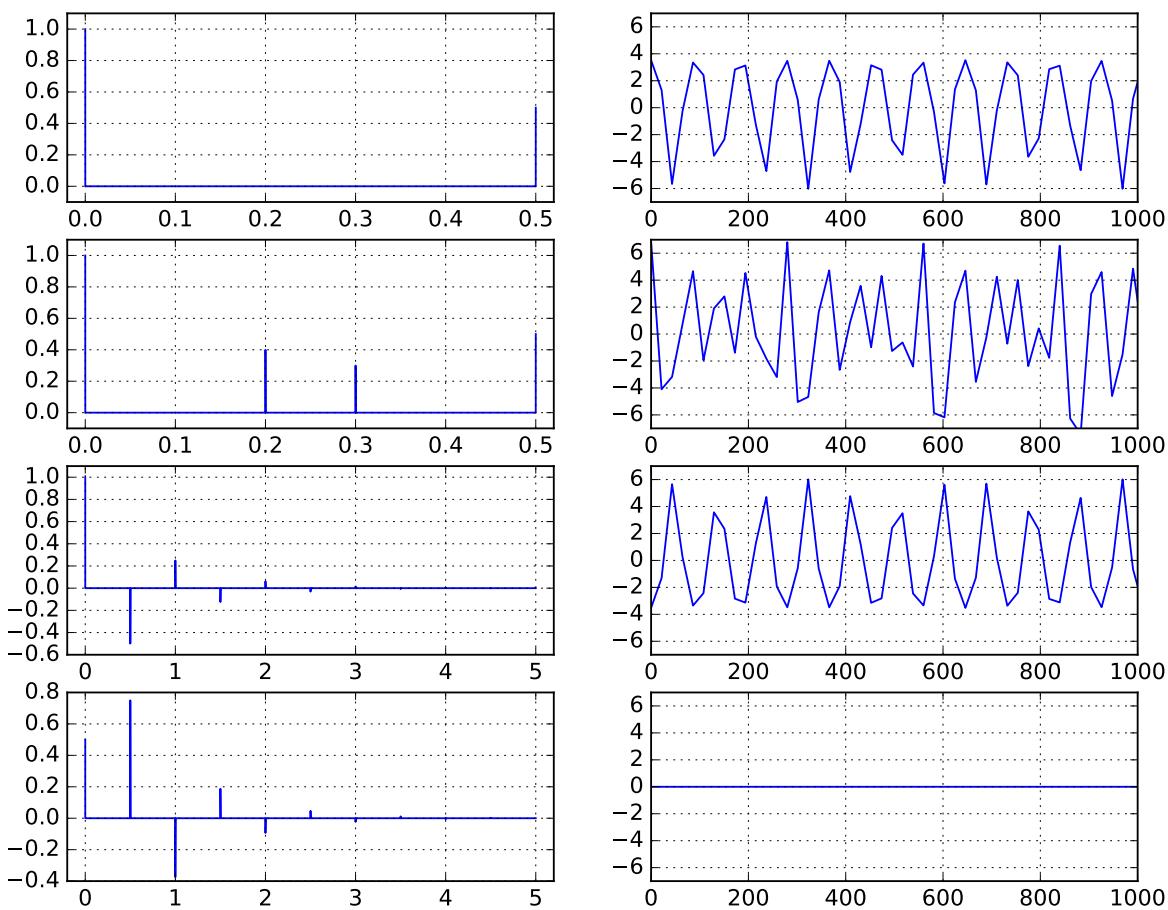
Следно ќе ги пресметаме импулсните и фреквенциските одсиви на овие четири системи.

```

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))

```

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



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.

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 #%% преслушување
66 os.system('play audio/Pato_22K.wav')
67 wavfile.write('audio/Pato_echo_fir.wav', fs, np.array(wav_echo, dtype='int16'))
68 os.system('play audio/Pato_echo_fir.wav')
69 wavfile.write('audio/Pato_echo_fir_mul.wav', fs, np.array(wav_echo, dtype='int16'))
70 os.system('play audio/Pato_echo_fir_mul.wav')
71 wavfile.write('audio/Pato_echo_iir.wav', fs, np.array(wav_echo, dtype='int16'))
72 os.system('play audio/Pato_echo_iir.wav')
73 wavfile.write('audio/Pato_echo_ap.wav', fs, np.array(wav_echo, dtype='int16'))
74 os.system('play audio/Pato_echo_ap.wav')

```

6.3 Реверберација

Реверберијата или ревербот претставува ефект во кој се генерираат и надодаваат многу повеќе задоцнети верзии од аудиосигналот со што се постигнува нивно аудиторно слевање во еден здружен оддек. Реверберијата е таа која му дава просторност на звукот, односно преку нејзе можеме да заклуччиме во каков вид на просторија го слушаме или е снимен звучниот сигнал. Реверберијата 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 **flenц**. 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.

flenц² 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 5.7, кој пак може да се реализира со каскада од променливи сепропусни филтри. Разликата меѓу двата ефекта е во тоа што фленцот 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 патентниот 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 алгоритамот. 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 менацирање 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 квантација 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 квантација 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

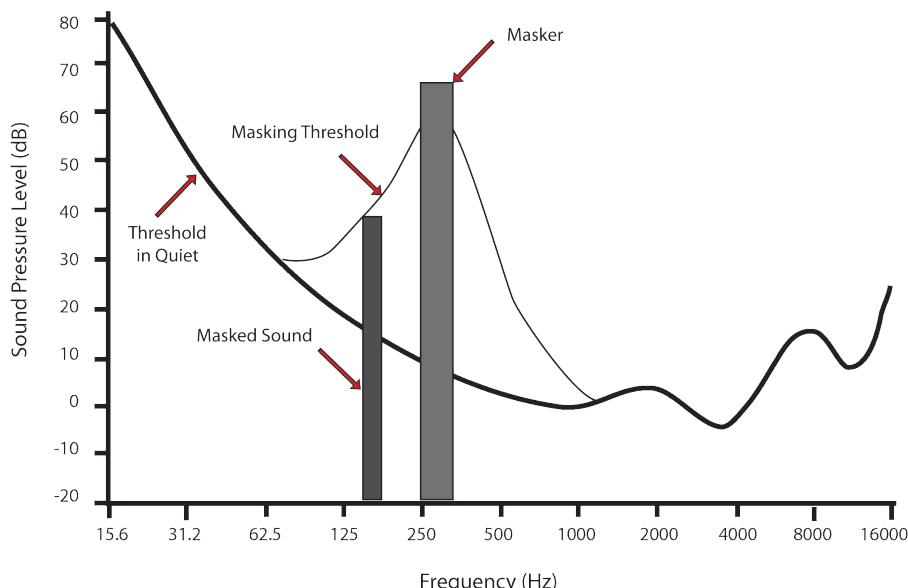
Притоа фреквенциското маскирање е тесно поврзано со **критичните опсези**⁵ на аудиторниот систем кои го одредуваат опсегот на фреквенции кои можат да бидат маскирани од една компонента во зависност од нејзината фреквенција. Тие всушност ја даваат ширината на пропусниот опсег на базилијарната мембра на функција 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 квантизација,
- анализа преку синтеза и
- ентрописко кодирање.

Доделувањето на битови и квантизацијата на MDCT спектралните линии се изведува преку вгнездена јамка која употребува и неуниформна квантизација и **Хофманово кодирање**. Внатрешната јамка го прилагодува чекорот на квантизација 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 битскиprotoци 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:

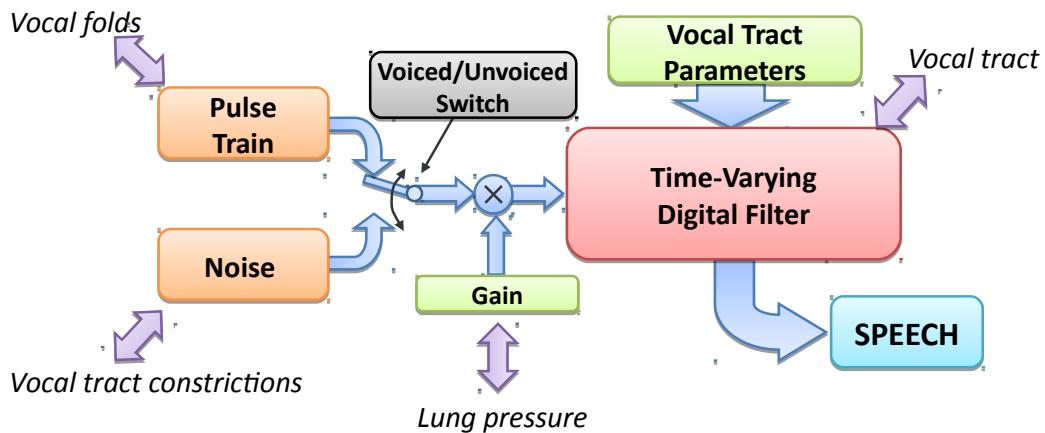
$$H(z) = \frac{Y(z)}{X(z)} = \frac{G}{1 - \sum_{k=1}^p a_k z^{-k}}, \quad (7.1)$$

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 запштеда во простор. Параметрите кои се екстрагираат за секоја рамка од говорниот сигнал се:

- звучност – дали се работи за звучен или беззвучен глас,
- засилување – енергијата на сигналот во рамката,



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

- **кофициенти 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, квалитетот на излезниот говор не е на високо ниво и има каркатегистичен 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.

⁷ $8 \text{ kHz} \cdot 8 \text{ bit} = 64 \text{ kbit/s}$

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 отсекоци 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.

Audacity

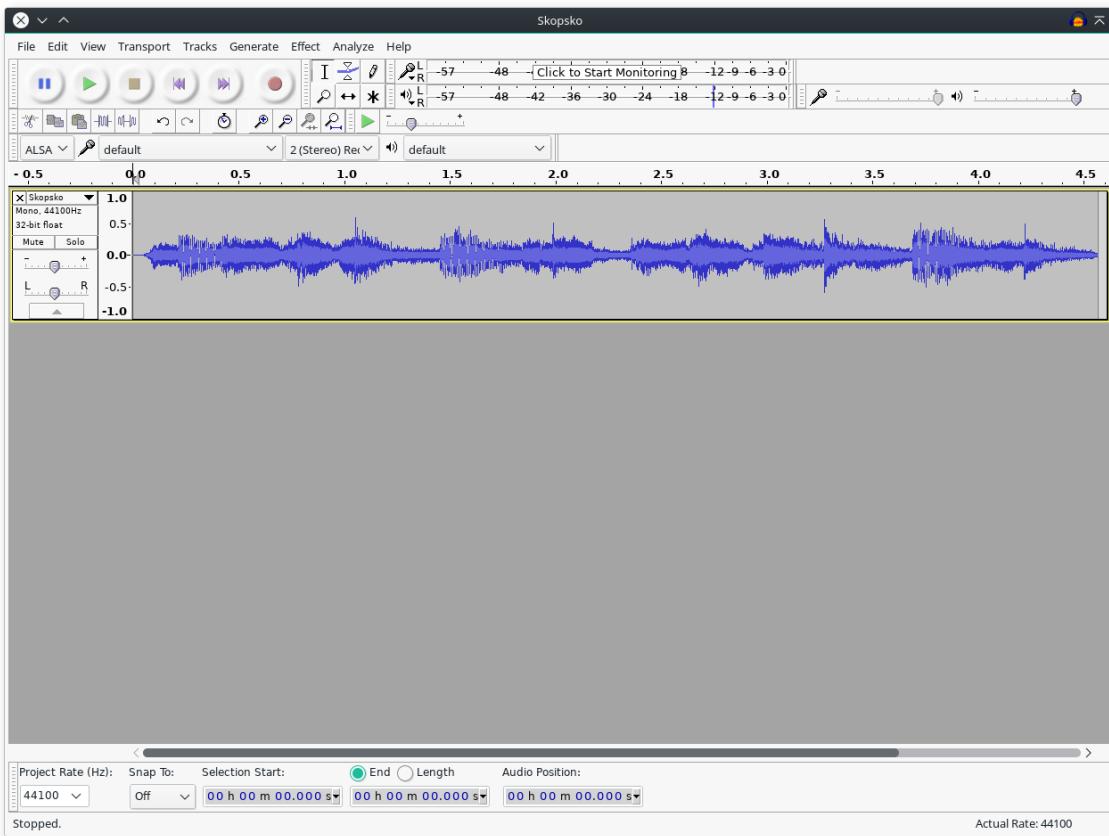
Audacity¹ е слободен софтвер за едитирање и снимање на дигитално аудио, достапен за сите оперативни системи, Сл. 8.1. Неговиот развој го започнале Доминик Мацони и Роџер Даненберг во 1999 во Универзитетот Карнеги Мелон и е иницијално објавен во 2000 како верзија 0.8. Од 2011, тој е 11-от најсимнуван 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.
- 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.

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))



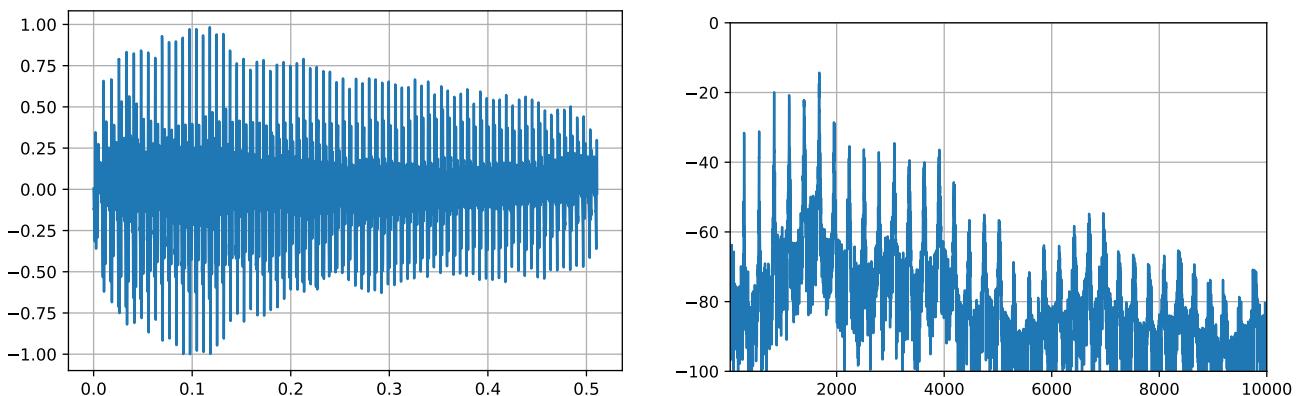
Сл. 8.1: Отворен аудио фајл во главниот прозорец на Audacity.

- Едитирање на амплитудната анвелопа.
- Намалување на шумот.
- Намалување на вокалите.
- Спектрална анализа со употреба на FFT.
- Поддршка 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.

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`:

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



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

```
$ 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.

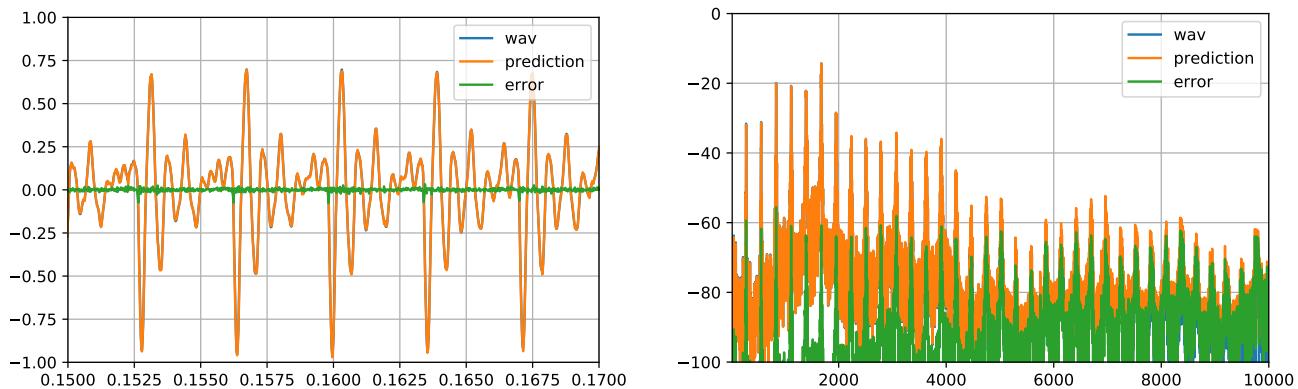
```
from __future__ import division
import numpy as np
from matplotlib import pyplot as plt
from scipy.io import wavfile
import os
from scipy import signal as sig
import das
from scikits.talkbox import lpc

#%%
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`.



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

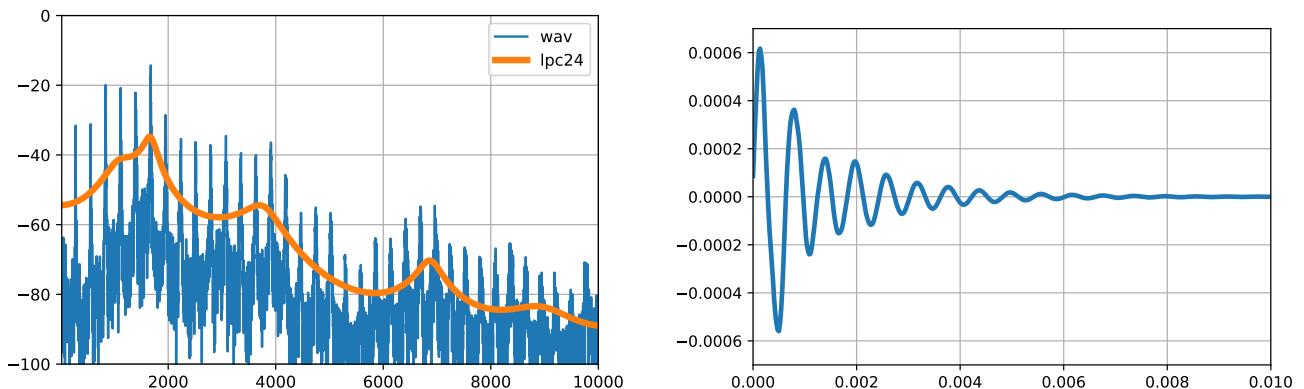
```
p = 24
a_filt, err, _ = lpc(wav, p)
b_inv = np.concatenate(([0], -a_filt[1:]))
wav_pred = sig.lfilter(b_inv, 1, wav)
wav_err = wav - wav_pred
```

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 хармониите 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.

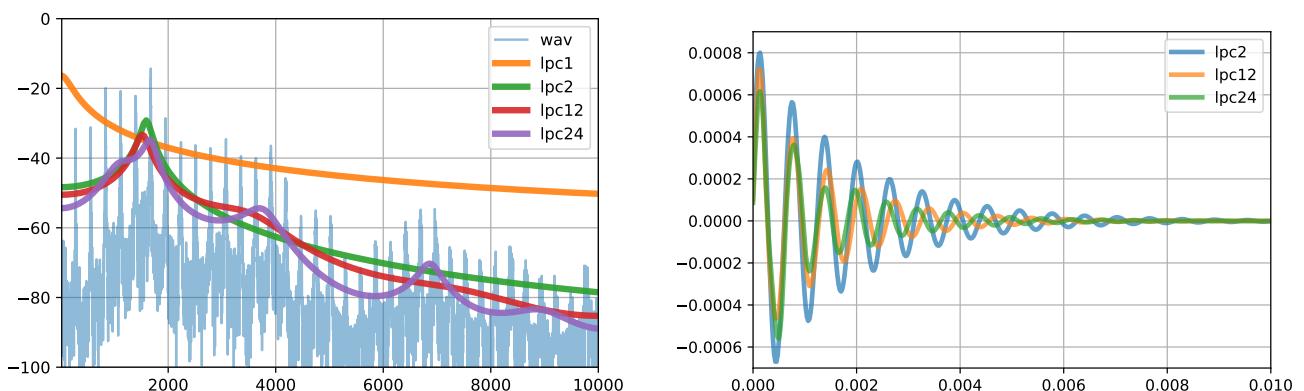
```
G = err
w, H_filt = sig.freqz(G, a_filt)
f_H = w / np.pi * fs/2
H_filt = 20*np.log10(np.abs(H_filt))
plt.figure()
plt.plot(f, wav_spec)
plt.plot(f_H, H_filt, lw=4)
```

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

⁴На англиски процесот на „беленење“ 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 /a/. 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.

Додаток А

Слободен и отворен софтвер за инженерска и научна работа

Еден од најпрочуените софтверски пакети за нумериичка обработка е програмскиот пакет **Матлаб¹**. Матлаб, преку својата синтакса на високо ниво дозволува: лесна манипулација на матрици, исцртување на функции и податоци, имплементација на алгоритми, создавање на кориснички интерфејси, итн. Тој може да се употреби во најразлични области од инженерската практика, меѓу кои и во дигиталната обработка на звук, слика и видео. За првпат бил издаден во 1984 г., а во 2004 г. имал 1 милион корисници инженери, научници и економисти.

Сепак Матлаб, како комерцијален софтвер носи и низа од недостатоци, пред сè високата цена која го става вон дофат на студентите, истражувачите, малите компании, како и на научно истражувачките и образовните установи во поголем дел од светот. Други недостатоци на Матлаб се ограничната преносливост на кодот, како и неговата затвореност.

A.1 Слободен софтвер

Денес сè повеќе инженери и научници ја напуштаат употребата на комерцијалниот затворен софтвер и својата работа ја засноваат на платформи базирани на **слободен софтвер²**. Ова пред сè се должи на философијата на движењето за слободен софтвер започнато од **Ричард Сталман³** во 1983 г. со креирањето на ГНУ оперативниот систем, а подоцна со воспоставување на Фондацијата за слободен софтвер⁴ во 1985 г., како и поширокото **движење за отвореност⁵**, а тоа е заедништво во создавањето и напредувањето на технологијата и човештвото.

A.2 Четири слободи

Слободниот софтвер е дефиниран со четирите слободи:⁶

- **Слобода 0.** Слобода да ја користите програмата за било која намена.

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

²Wikipedia – Free software movement https://en.wikipedia.org/wiki/Free_software_movement

³Wikipedia – Richard Stallman https://en.wikipedia.org/wiki/Richard_Stallman

Предавање на Ричард Сталман за философијата на движењето за слободен софтвер – Richard Stallman – Free software, free society, TEDxGeneva 2014 https://www.youtube.com/watch?v=Ag1AKI1_2GM

⁴Wikipedia – Free Software Foundation https://en.wikipedia.org/wiki/Free_Software_Foundation

⁵Wikipedia – Open-source model https://en.wikipedia.org/wiki/Open-source_model

Nathan Seidle – How Open Hardware will Take Over the World, TEDxBoulder https://www.youtube.com/watch?v=xGhj_1LNtd0

⁶Превземено од вебстраницата на организацијата Слободен софтвер Македонија <https://slobodensoftver.org.mk/shto>

Добавањето рестрикции за користење на слободен софтвер, како што се временските рестрикции („Пробен период од 30 дена“, „Лиценцата истекува на 1 јануари 2005“), рестрикции на целта („Дозволена е употреба за истражувачки и некомерцијални цели“) или рестрикции на географската област („Мора да се користи во земјата А“), ја прават програмата неслободна.

- **Слобода 1.** Слобода да проучите како работи програмата и како истата да ја адаптирате на сопствените потреби.

Добавањето легални или практични рестрикции на разбирањето или менувањето на програмата, како што се задолжително купување на специјални лиценци, потпишување на спогодба за неоткривање (Non-Disclosure-Agreement) или правењето изворниот код да биде недостапен, исто така ја прават програмата неслободна. Без слободата да се менува програмата, луѓето ќе останат на милост на единствен снабдувач.

- **Слобода 2.** Слобода да редистрибуирате копии за да му помогнете на вашиот сосед.

Софтверот може да се копира/дистрибуира скоро без никакви трошоци. Ако не смеете да му дадете некоја програма на некој човек кому таа му треба, тоа ја прави програмата неслободна. Се разбира, доколку сакате, за ваквите активности можете да наплатите.

- **Слобода 3** Слобода да ја подобрувате програмата и да ги издадете вашите подобрувања во јавноста, од што корист ќе има целата заедница.

Сите луѓе не се подеднакво добри програмери. Некои луѓе пак воопшто не знаат да програмираат. Оваа слобода им дозволува на оние луѓе кои немаат време или знаење да решат некој проблем индиректно да пристапат до слободата за менување на програмата. Се разбира, доколку сакате, за ваквите активности можете да наплатите.

Доколку софтверот не ги исполнува сите горни услови, тогаш тој не е слободен софтвер.

A.3 Предности на слободниот софтвер

Од практичен аспект, отворениот софтвер има низа предности над затворениот софтвер и тоа:

- **достапноста** – 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, но и од продавање поддршка за нивниот производ.
- **безбедноста** – поради достапноста на изворниот код, не постои начин производителот на софтверот 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.^{7,8}
- **слободата од производителот** – како корисници на отворениот софтвер, вие не сте затворени во екосистемот на производителот.⁹ Истиот тој софтвер може да биде превземен од друга заедница на програмери и да продолжи неговото одржување и развој во друга насока.

⁷ Во Windows 10 производителот го задржува правото да ги чува вашите приватни податоци како што вели 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 ...”

Истите механизми се додадени во претходните верзии на Windows преку автоматските надградби.

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-spying-how-to-opt-out/>

Ashley Allen, How to Stop Windows 7 and 8 From Spying on You <http://www.eteknix.com/stop-windows-7-8-spying/>

⁸ Епл и Самсунг ги забавија телефоните на корисниците преку нивното редовно ажурирање <https://www.cnet.com/news/apple-and-samsung-fined-for-slowing-down-phones-with-updates/>

⁹ Don Reisinger – Steve Jobs wanted to ‘further lock customers’ into Apple’s ‘ecosystem’ <https://www.cnet.com/news/steve-jobs-wanted-to-further-lock-customers-into-apples-ecosystem/>

- подобар 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 јадрото¹⁰, кое е во основата оперативниот систем ГНУ/Линукс познат и само како Линукс¹¹ и повеќе од 600-те ГНУ/Линукс дистрибуции¹², првично напишано од Линус Торвалдс¹³, денес претставува најголемиот здружен проект во историјата на човештвото со околу 6000 активни развивачи, над 20 милиони редови на код, и со проценета развојна вредност од над 2 милијарди евра.¹⁴

Сите овие придобивки заедно придонесуваат за широка рас пространетост на слободниот софтвер денес. Така, ГНУ/Линукс и ФриБСД¹⁵ оперативните системи се во употреба во 98,27 % од серверите на интернет (споредено со 1,73 % со Виндоус), 79,3 % од паметните телефони (Андроид оперативниот систем)¹⁶, и 99 % од суперкомпјутерите¹⁷. Сепак, неговиот пробив во персоналните компјутери засега е незначителен – 2,1 % (наспроти 87 % на Виндоус и 9,7 % на МекОС).

A.4 Одржливост

Постојат различни начини на кои се реализира финансиската поддршка на слободниот софтвер и покрај бесплатноста и тоа:

- финансиска поддршка од компанији – зад многу пакети слободен софтвер стојат компанији од чиј интерес е неговиот развојот, Најдобар пример за тоа е можеби самото Линукс јадро на кое работат инженери од многу компанији од целиот свет, а најголемиот придонес го има компанијата Интел. Тука се и низа на ГНУ/Линукс дистрибуции меѓу кои Убунту¹⁸, Федора¹⁹, и ОпенСусе²⁰, како и пакетите за длабоко учење Тензорфлоу²¹ и Пајторч²², исто така развивани од компанији,
- финансиска поддршка од јавно финансирање и грантови – голем број на слободни софтвери се плод на работата на инженери и научници финансиирани од државите низ светот или од приватни фондации. Таков е на пример КиКАД софтверот за електронски дизајн и изработка на печатени плочи развибан во ЦЕРН²³, софтверот за обработка на аудио Аудасити започнат во Универзитетот Карнеги Мелон²⁴, или пак пакетот за машинско учење Сајкитлрн започнат во Инриа²⁵,
- бизнис модел базиран на поддршка – најголемата компанија која денес работи исклучиво со слободен софтвер е Ред Хет чиј ГНУ/Линукс оперативен систем е еден од најзастапените на интернет серверите.²⁶ Ред Хет заработува преку продажба на поддршка за овој оперативен систем и во моментов е проценета на вредност од 38 милијарди УСД,
- финансиска поддршка од донацији – многу слободни софтвери егзистираат благодарејќи на донацији направени од нивните корисници. Тука спаѓаат најголем број од ГНУ/Линукс

¹⁰ Wikipedia – Linux kernel https://en.wikipedia.org/wiki/Linux_kernel

¹¹ Wikipedia – Linux <https://en.wikipedia.org/wiki/Linux>

¹² Wikipedia – List of Linux distributions https://en.wikipedia.org/wiki/List_of_Linux_distributions

¹³ Wikipedia – Linus Torvalds https://en.wikipedia.org/wiki/Linus_Torvalds

¹⁴ Добар документарен филм за рафањето и развојот на ГНУ/Линукс оперативниот систем е Revolution OS - 2001 <https://www.youtube.com/watch?v=Eluzi700-P4>

¹⁵ FreeBSD <https://www.freebsd.org/>

¹⁶ Оваа бројка е речиси 91 % ако се има в' предвид дека и iOS е базиран на Линукс јадрото.

¹⁷ Linux is Running on Almost All of the Top 500 Supercomputers <https://itsfoss.com/linux-supercomputers-2017/>

¹⁸ Ubuntu <https://www.ubuntu.com/>

¹⁹ Fedora <https://getfedora.org/>

²⁰ OpenSUSE <https://www.opensuse.org/>

²¹ TensorFlow – An end-to-end open source machine learning platform <https://www.tensorflow.org/>

²² PyTorch – from research to production <https://pytorch.org/>

²³ KiCad EDA – A Cross Platform and Open Source Electronics Design Automation Suite <http://kicad-pcb.org/>

²⁴ Audacity – Free, open source, cross-platform audio software <https://www.audacityteam.org/>

²⁵ scikit-learn – Machine Learning in Python <https://scikit-learn.org/stable/index.html>

²⁶ Red Hat – The world's leading provider of open source solutions <https://www.redhat.com>

дистрибуциите како на пример Манџаро²⁷ или Минт²⁸, а исто така СпајдерSpyder – The Scientific Python Development Environment <https://manjaro.org/> развојната средина за Питон која ќе ја користиме во предметов,

- ентузијазам – мотивот нешто да се создаде или подобри и да се сподели со целиот свет понекогаш е доволен мотив за развој на слободниот софтвер. Постојат низа пакети со заедници на развивачи кои немаат финансиски придобивки од нивната работа на проектот, но сепак продолжуваат да работат на него водени од сопствените убедувања и стремеж кон повисоки вредности.

A.5 Слободен софтвер за инженерска и научна работа

Постојат низа на слободни софтвери кои можат да бидат искористени за обработка на нумерички податоци.

- ГНУ Октејв²⁹ има синтакса направена да биде во голема мера компатибилна со онаа на Матлаб. Во Октејв се реализирани голем број на пакети кои можат да се искористат за обработка на најразлични типови на сигнали. Проблемот со Октејв е во неговата мала брзина на извршување, поради што највеќе се употребува во образоването како замена за Матлаб.
- Сајлаб³⁰ е слободен софтвер за нумеричка обработка наменет за инженери и научници, во употреба од 1994 г. Сајлаб во себе вклучува и слободна замена за Симулинк пакетот на Матлаб, наречена Икскос³¹.
- Питон³² е широко распространет, повеќенаменски, интерпретиран и динамичен програмски јазик на високо ниво направен од Гуидо ван Росум³³ во 1989 г. Иако не е наменет строго за нумеричка анализа, елегантната и едноставна синтакса која овозможува лесна читливост, како и неговата широка распространетост во најразлични области, го прават Питон идеална основа за слободната работа и соработка на научната и образовната заедница ширум светот.
- Џулиа³⁴ е јазик за нумеричко процесирање со компајлирање направен на МИТ, кој иако има синтакса на високо ниво како онаа на Матлаб, работи речиси еднакво брзо со код напишан во С. И покрај големиот потенцијал на Џулија, за сега неговата примена останува ограничена во области во кои е неопходна голема процесирачка моќ.

²⁷Manjaro – Professional Linux at its best <https://manjaro.org/>

²⁸Linux Mint – From freedom came elegance <https://linuxmint.com/>

²⁹GNU Octave – Scientific Programming Language <https://www.gnu.org/software/octave/>

³⁰Scilab – Open source software for numerical computation <https://www.scilab.org/>

³¹Xcos <https://www.scilab.org/software/xcos>

³²Python <https://www.python.org/>

³³Wikipedia – Guido van Rossum https://en.wikipedia.org/wiki/Guido_van_Rossum

³⁴The Julia Programming Language <https://julialang.org/>

Додаток А

Питон за процесирање на аудиосигналите

За процесирањето на дигиталните аудиосигнали ќе биде искористен програмскиот јазик Питон и тоа неговата нова верзија 3, заедно со библиотеките:

- Нумпай – за работа со вектори и матрици,¹
- Сајпај – за дигитално процесирање на сигнали,²
- Матплотлиб – за визуелизација.³

Освен овие постојат мноштво библиотеки за Питон кои се користат во научните истражувања како на пример **Пандас**⁴ за статистички анализи, **Симпај**⁵ за симболичка математика, **Сајкитлрн**⁶ за машинско учење итн.

Како интерфејс кон Питон ќе ја користиме интерактивната конзола **ИПитон**⁷ и научната развојна средина за Питон **Спајдер**⁸.

A.1 Основи поставки во ГНУ/Линукс

Иако користењето на Питон не е врзано со ГНУ/Линукс оперативниот систем, вежбите во овој предмет ќе се базираат на работа под ГНУ/Линукс. Доколку веќе немате ГНУ/Линукс, истиот се препорачува да го инсталirate паралелно на постоечкиот оперативен систем. Во најмала рака може да инсталirate ГНУ/Линукс во виртуелна машина, но ова може да ги ограничи постоечките ресурси за процесирање на сигналите.⁹ Во Лабораторијата за дигитално процесирање на сигнали ќе работиме со **Убунту Мате**¹⁰ кој е базиран на **Убунту**, а ја користи десктоп средината **Мате** чиј изглед е базиран на **Гном 2**, а е имплементиран во **Гном 3**.¹¹

¹NumPy <http://www.numpy.org/>

²SciPy <http://www.scipy.org/>

³Matplotlib <http://matplotlib.org/>

⁴Pandas <http://pandas.pydata.org/>

⁵Sympy <http://www.sympy.org/en/index.html>

⁶SciKit-Learn <http://scikit-learn.org/stable/>

⁷IPython Interactive Computing <http://ipython.org/>

⁸Spyder – The Scientific PYthon Development EnviRonment <https://github.com/spyder-ide/spyder>

⁹Добар преглед на популарноста на различните Линукс дистрибуции, како и повеќе информации за истите може да најдете на вебстраницата *Distrowatch*. <http://distrowatch.com/>

¹⁰Ubuntu MATE. <https://ubuntu-mate.org/>

¹¹Кај ГНУ/Линукс оперативните системи можат да се користат различни десктоп средини како GNOME и MATE, но уште и KDE Plasma, Xfce, LXQt, Cinnamon, Pantheon, итн.

За работа со ГНУ/Линукс можеме да го искористиме стандардниот БАШ терминал.¹² Вообичаена кратенка за отворање на нов терминал е `ctrl-alt-t`, или ако дистрибуцијата доаѓа со терминал на спуштање копчето `F12`.

За почеток треба во вашиот основен фолдер¹³ да отворите нова папка со името на предметот. Тоа може да го направите преку фајл експлорерот, или преку терминалот:

```
~ $ mkdir das
```

Следно, од Гитхаб страната на предметот Дигитални аудиосистеми¹⁴ превземете го фолдерот со звучни сегменти кои ќе ги користиме во вежбите. Тоа можете да го направите на следниот начин:

```
~ $ cd das
~/das/ $ git clone https://github.com/FEEIT-FreeCourseWare/Digital-Audio-Systems.git
~/das/ $ cp -r Digital-Audio-Systems/code/audio .
```

За да може да ги слушнеме овие аудиозаписи треба да го инсталлираме `SoX`¹⁵ кој претставува моќна алатка за конверзија на аудиофајлови од еден формат во друг, но може да се искористи и за додавање на различни аудиоэффекти, како и снимање и преслушување на аудиофајлови. За инсталирање и надградба на софтверот и самиот оперативен систем во Линукс е одговорен менаџерот на пакети. Кај дистрибуциите од фамилијата на Убунту како менаџер се користи `apt-get` или на повисоко ниво `apt`. Поради безбедносни причини во Линукс при секое менување на инсталираниот софтвер и системските фајлови мора да се повикаме на администраторски привилегии преку наредбата `sudo`¹⁶:

```
~/das/ $ sudo apt install sox
```

По што може да преслушаме некој од аудиофајловите:

```
~/das/ $ play audio/Solzi.wav
```

`Solzi.wav:`

```
File Size: 345k      Bit Rate: 714k
```

```
Encoding: Signed PCM
```

```
Channels: 1 @ 16-bit
```

```
Samplerate: 44100Hz
```

```
Replaygain: off
```

```
Duration: 00:00:03.87
```

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

A.2 Основи на работата со Питон

Питон интерпретер

За работа со Питон може да ја искористиме стандардната инсталацијата на Питон која доаѓа со секоја ГНУ/Линукс дистрибуција. Питон интерпретерот можеме да го повикаме во стандардниот

¹²Bourne-Again Shell (BASH) средината е стандардна за сите дистрибуции на ГНУ/Линукс и МекОС. Добро напатство за нејзина употреба претставува: Software Carpentry – The Unix Shell <http://swcarpentry.github.io/shell-novice/>

Постојат и понапредни шел средини како на пример Z shell – скратено Zsh. Еве еден туторијал за нејзина инсталација: Oh-My-Zsh! A Work of CLI Magic—Tutorial for Ubuntu <https://medium.com/wearetheledger/oh-my-zsh-made-for-cli-lovers-installation-guide-3131ca5491fb>

¹³Бо ГНУ/Линукс основен фолдер на секој корисник е `/home/user_name/`, а кратенка за него е `~`.

¹⁴<https://github.com/FEEIT-FreeCourseWare/Digital-Audio-Systems>

¹⁵SoX - Sound eXchange. <http://sox.sourceforge.net/>

¹⁶Кратенка од *super user do*.

БАШ терминал со:

```
$ python

Python 3.7.2 (default, Jan 10 2019, 23:51:51)
[GCC 8.2.1 20181127] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('hello world')
hello world
```

За излегување од Питон конзолата треба да ја притиснеме стандардната кратенка `ctrl-d` или да напишеме `exit()`.

ИПитон

Поради ограничените можности на основниот Питон интерпретер, вообичаено со Питон се работи во интерактивната конзола **ИПитон** која нуди низа на подобрувања. Нејзe може да ја инсталлираме со:

```
$ sudo apt install ipython
```

а по инсталацијата може да ја повикаме со:

```
$ ipython
```

```
Python 3.7.2 (default, Jan 10 2019, 23:51:51)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.3.0 -- An enhanced Interactive Python. Type '?' for help.

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

Некои од главните придобивки кои ги носи ИПитон се:

- пристап до стандардната помош во Питон, како на пример докстрингови на објекти и напатствието за Питон, преку наредбата `help`,
- низа од специјални наредби, наречени и „магии“, како на пример `%timeit` за мерење на времето потребно за извршување на една наредба, `%matplotlib` за овозможување на интерактивно исцртување, или `%history` за испишуваче, пребарување или запишување на историјата на извршените наредби; повеќе за овие наредби може да се види со наредбата `%magic`,
- информации за секој објект преку употреба на `?` наредбата,
- автоматско комплетирање на имињата на објектите и променливите од локалниот простор на имиња, како и имиња од локалниот фолдер, со употреба на `Tab` копчето,
- пребарување на претходно внесени наредби со стрелките и внесување на првите букви од саканата наредба, а со `ctrl-r` и со пребарување на целата содржина на претходните наредби,
- извршување на шел наредби со помош на `!` .

За да ги видите сите можности кои ги нуди ИПитон напишете `?` или `%quickref` во интерактивната конзола.

Дополнително, ИПитон е основата зад **Јупајтер Кјутконзолата**¹⁷ прикажана на Сл. А.1, која е реализирана во Кјут технологијата и овозможува плотирање во самата конзола кое

¹⁷Jupyter QtConsole <https://github.com/jupyter/qtconsole>

```

File Edit View Kernel Window Help
Jupyter QtConsole 4.1.1
Python 2.7.10 (default, Oct 14 2015, 16:09:02)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.3 -- 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.
%uiref     -> A brief reference about the graphical user interface.

In [1]: from __future__ import division
...: import numpy as np
...: from scipy.io import wavfile
...: from matplotlib import pyplot as plt
...:

In [2]: %matplotlib inline

In [3]: plt.plot(np.arange(10))
Out[3]: []

9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9
In [3]:

```

Сл. А.1: Јупајтер Кјут конзолата нуди напредна интерактивност.

може да се активира со наредбата `%matplotlib inline`. Таа се стартира во терминалот со наредбата:

```
$ jupyter qtconsole
```

Виртуелни средини за Питон

Инсталирањето на Питон пакети, како што направивме со ИПитон козолата, директно во оперативниот систем не е препорачливо. За избегнување на судир помеѓу системската инсталација на Питон на ГНУ/Линукс оперативниот систем, како и за изолирање на екосистемот од инсталирани модули на секој засебен проект, правилно е да направиме Питон виртуелна средина за процесирање на дигиталните аудиосигнали. Постојат низа пакети за создавање и раководење со виртуелните средини во Питон, од кои `pipenv`¹⁸ е оној кој е препорачан од Телото за пакување на Питон¹⁹.

За инсталирање на `pipenv` треба да го инсталерираме користејќи го системскиот `pip`²⁰. Поради можноста на употреба на Питон 2 како стандарден во инсталираната ГНУ/Линукс дистрибуција, најдобро е да напишеме:

```
$ sudo pip3 install pipenv
```

А доколку системот го нема `pip3`, истиот може да се инсталира како:

¹⁸Pipenv – Python Development Workflow for Humans <https://github.com/pypa/pipenv>

¹⁹Python Packaging Authority PyPA <https://www.pypa.io/en/latest/>

²⁰Python Install Package (pip) <https://pypi.org/project/pip/>

```
$ sudo apt install python3-pip
```

За креирање на виртуелна средина во фолдерот за овој предмет ќе напишеме:

```
$ mkdir das
$ pipenv --python 3
```

```
Creating a virtualenv for this project...
Pipfile: /tmp/das/Pipfile
Using /usr/bin/python3 (3.7.2) to create virtualenv...
Creating virtual environment...Using base prefix '/usr'
New python executable in ~/.local/share/virtualenvs/das-aWLbLjVf/bin/python3
Also creating executable in ~/.local/share/virtualenvs/das-aWLbLjVf/bin/python
Installing setuptools, pip, wheel...
done.

Running virtualenv with interpreter /usr/bin/python3

Successfully created virtual environment!
Virtualenv location: ~/.local/share/virtualenvs/das-aWLbLjVf
Creating a Pipfile for this project...
```

Со ова pipenv креира нова виртуелна средина во локален фолдер во нашиот кориснички фолдер, во неа го копира системскиот Питон 3 и pip. За активирање на виртуелната средина може да ја искористиме наредбата pipenv shell која ја става патеката на виртуелната средина како прва во листата на системски патеки. Така следниот пат кога ќе сакаме да го повикаме Питон интерпретерот или да инсталураме пакет со pip, тоа ќе се случува внатре во виртуелната средина:

```
$ which pip
/usr/bin/pip

$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/bin ...

$ pipenv shell
Launching subshell in virtual environment...
. ~/.local/share/virtualenvs/das-aWLbLjVf/bin/activate

(das) $ echo $PATH
~/.local/share/virtualenvs/das-aWLbLjVf/bin:/usr/local/sbin:/usr/local/bin:...

(das) $ which pip
~/.local/share/virtualenvs/das-aWLbLjVf/bin/pip
```

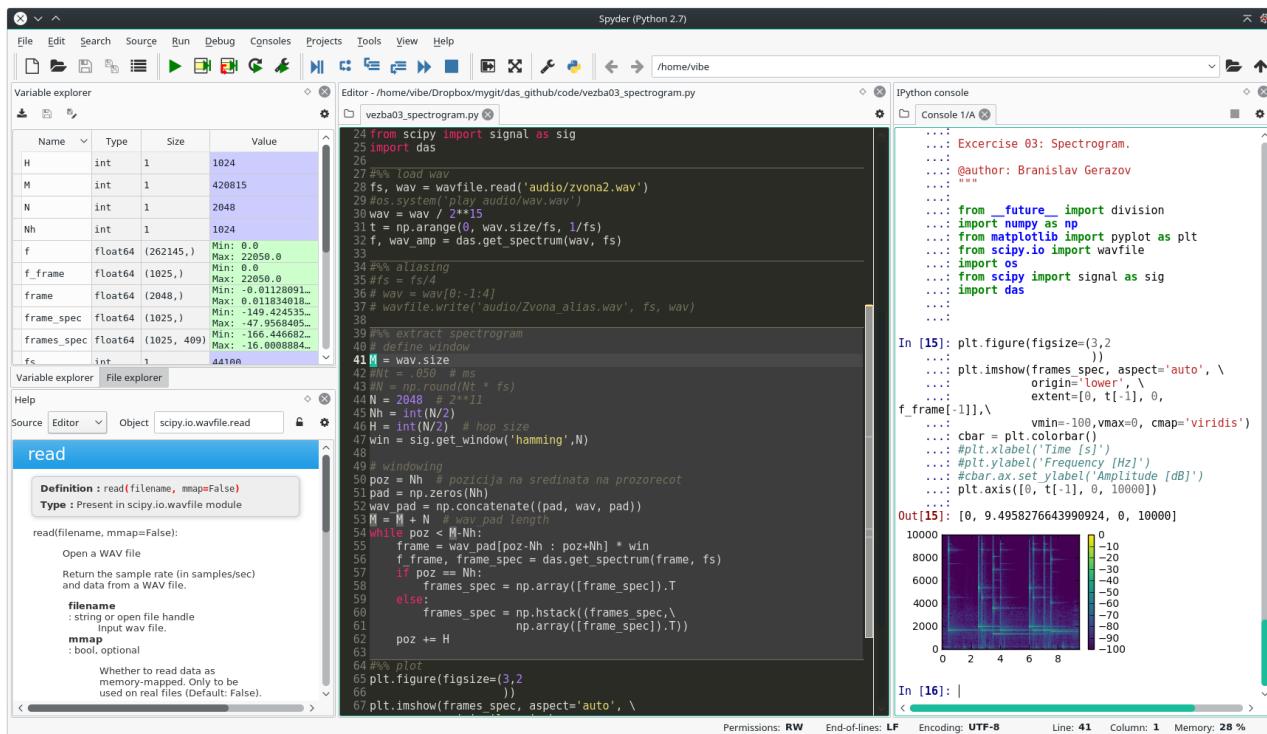
Гледаме дека pipenv ни дозначува дека сме во виртуелната средина со (das) пред БАШ промптов. За да излеземе од неа повторно може да ја употребиме кратенката ctrl-d или да напишеме exit.

Спајдер

Наместо да работиме со ИПитон конзолата во терминалот, во предметот дигитални аудиосистеми ќе ја употребиме развојната средина за Питон специјализирана за инженерска и научна работа Спајдер²¹ прикажана на Сл. А.2. Спајдер во себе вклучува:

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

²¹Spyder - The Scientific PYthon Development EnviRonment. <https://github.com/spyder-ide/spyder>



Сл. А.2: Спајдер развојната средина за Питон специјализирана за инженерска и научна работа.

- **Интерактивна конзола** – интегрирани Питон и иПитон конзоли со работни простори и подршка за дебагирање и поддршка за Матплотлиб, 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.**

Спајдер можеме да го инсталлираме во виртуелната средина која ја креирајме во работниот фолдер `das`:

```
(das) $ pip install spyder
```

A.3 Основи на Нумпај и Матплотлиб

За да ги инсталлираме потребните модули во новата виртуелна средина ќе напишеме:

```
(das) $ pip install numpy matplotlib scipy
```

Сега може да ги повикаме во ИПитон:

```
In [1]: import numpy as np
In [2]: from matplotlib import pyplot as plt
In [3]: x = np.linspace(0, 2*np.pi, 100)
In [4]: y = np.sin(x)
In [5]: %matplotlib
```

```
Using matplotlib backend: Qt5Agg

In [6]: plt.plot(x, y)
Out[6]: [

```

Забележете дека го импортираме Нумпај со `import numpy as np` наместо со `from numpy import *`. Ова е препорачана практика за избегнување на оптеретување на постоечките функции во основниот простор на имиња²², како и за зачувување на засебен простор со имиња за секој од импортирани модули. Ова овозможува и одлично автоматско надополнување на започнатото име на модул, функција или променлива во Питон.²³ Со наредбата `%matplotlib` се овозможува интерактивното плотирање во ИПитон конзолата.

Добар вовед во програмскиот јазик Питон во рамки на екосистемот за научна работа е даден во Скриптата за Сајпаж (Varoquaux et al., 2015)²⁴ која е достапна под слободна лиценца²⁵. Оваа книга претставува отворен проект и во неа, благодарејќи на многуте автори и придонесувачи, се поместени основите за работа не само со Питон, Нумпај, Матплотлиб и Сајпаж, туку и Пандас, Симпаж, Сајкитимиџ, Сајкитлрн, па дури и Џајтон²⁶.

За запознавање со основите на програмскиот јазик Питон можат да послужат мноштво на материјали, каков што е официјалниот туторијал за Питон на вебстраницата на Питон²⁷ и Викикнигата Програмирање во Питон²⁸, обете во употребува на МИТ.

²²Добрата структурираност на просторите на имиња (namespaces) е една од важните одлики на Питон како што е и наведено во стиховите на Зенот на Питон кој може да го прочитате ако напишете `import this`.

²³Автоматското надополнување се активира со притискање на `Tab`.

²⁴Scipy Lecture Notes <http://scipy-lectures.org/>

²⁵Creative Commons <https://creativecommons.org/>

²⁶Cython <https://cython.org/>

²⁷The Python Tutorial. <https://docs.python.org/3/tutorial/index.html>

²⁸Python Programming https://en.wikibooks.org/wiki/Python_Programming

Литература

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

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.

Gael Varoquaux, Valentin Haenel, Emmanuelle Gouillart, Zbigniew Jędrzejewski-Szmek, Ralf Gommers, Fabian Pedregosa, Olav Vahtas, 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/>.

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