ДОКУМЕНТАЦИЯ НА ПРОЕКТ

по Цифрова Обработка на Сигнали

на Деян Василев Добромиров фн 011331024

ръководител гл. ас. д-р Г. Ружеков/………./

Тема: Алгоритми за трансформация на Фурие

Последователността на работата за подкарване на проекта е следната:

1. Инсталиране на МАТЛАБ ( ако не е инсталиран )
2. Създава се директория „\toolbox\mp3\_toolbox\_v2.0“
3. Разработване на собствено програми за плейъра като

wipe – изчиства АБСОЛЮТНО всичко – работи като отделен m-фаил, който не е функция. След извикване затваря всички отворени фигури, изтрива променливите и показва нов, празен екаран в конзолата.

shiftArr – премества колона от стойности на матрица наляво, надясно – работи като функция, на която се предават параметри като входната матрица, която трябва са се премести, и начин по който да я пемести. Начинът се определя като цяло число.

Пример: a = [1 2;3 4];

shiftArr(a,-1) ans = [2 0; 4 0]

shiftArr(a, 1) ans = [0 1; 0 3]

swaparr – разменя стойностите на матрица по редове или колони, така че първия да стане последен, последния пръв и т.н. – работи като функция, на която се предават параметри като входната матрица, която трябва са се размени, и начин по който да я размени

Пример: a = [1 2;3 4];

swaparr(a,1) ans = [2 1; 4 3]

swaparr(a,2) ans = [3 4; 1 2]

mp3rate – променя честотата на дискретизация на обект от тип “audioplayer” – работи като функция, на която се предават параметри като обекта за манипулация и числото което трябва да се добави към настоящата честота на дискретизация. Функцията има грижата да спре възпроизвеждането, добави нужното чило към текущата честота на дискретизация, след което да породължи възпроизвеждането.

percent\_q – смята процент от число и връща процента – работи като функция, на която се предават параметри, като базовото число и число по-малко от него. След изчислението се връща резултат съответстващ на това, каква част ( в % ) е малкото число от голямото

Пример: percent\_q(25, 51) ans = 49.0196

part\_q – смята процент от число и връща числото – работи като функция, на която се предават параметри, като базовото число и процента от него който трябва да бъде изчислен. Резултата е число с плаваща запетая, което представлява предадения процент от базовото число

Пример: part\_q(33.3333,120) ans = 40.0000

int2bits – превръща число в неговия двоичен еквивалент– работи като функция, на която се предава желаното число и дължината на разрядната решетка, след което връща масив от нули и единици. Използва познатия на всички алгоритъм за конвертиране на числа от сесетична в двоична бройна система.

Пример: int2bits(13,5) ans = [ 0 1 1 0 1 ]

bits2int – обратно на int2bits – работи като функция, на която се предава входен масив от двоичен еквивалент и връща десетичното число

Пример: bits2int([1 0 1 0]) ans = 10

getbit2 – изчислява колко бита ще са необходими за кодиране на дадено число – работи като функция, на която се предава даденото положително число, след което се изчислява логаритъма му при основа 2. Резултата се закръгля и се прибавя 1.

Пример: getbit2(10) ans = 4

GetBitLen – конвертира число в двоичен код – работи като int2bits, но само за положителни числа или нула

getfpow – намива степента на WN за съответната фаза и индекс на масива А – работи като функция, на която се предават параметри, като текущия индекс на масива А и управляващия бит на съответната фаза

Пример: getfpow(3,2) ans = 2

getW - изчислява стойността на WN за съответната степен по познатата ни от трансформацията на Фурие формула – работи като функция, на която се предават параметри, като фиксираните до степен на 2-ката брой точки върху 2 и степента на Wn

Пример: getW(2,8) ans = 0.7071 - 0.7071i

Wn - изчислява стойността на WN вектор – работи като функция, на която се предава броя точки за изчисление на трансформацията на Фурие

Пример: Wn (4) ans = [ 1, -i ]

zerofil – допълва масив с нули до по-големия индекс от текущия, кратен на степен на двойката – работи като функция, на която се предава входния масив, недопълнен до степен на двойката. Функцията изчислява дължината на масива и проверява дали е изпълнено условието – дали побитов И от дължината и дължината – 1 и 1 е нула. Ако да, тигава масива е с дължина степен на 2 и просто го връща. Иначе изчислява оставащия брой нули които трябва да се добавят, като 2 на степен целочислена стойност от логаритъм при основа 2 от дължината + 1 , след което добавя 1 и ползва полученият резултат за да добави толкова нули към масива.

Пример: zerofil([1 2 3 4 5]) ans = [ 1 2 3 4 5 0 0 0 ]

lastslash – намира индекса на последната наклонена чесрта в стринг, като извърта цикъл със стъпка -1 – работи като функция, на която се предава входния масив от синволи, а резултата е число-индекс, съответсващо на първия символ от кадето започва стринга, който не съдържа наклонени черти.

Пример: lastslash('as\asf') ans = 4

mirnum2 – битова огледалност на число за даден брой битове – работи като функция, на която се предава числото, чиито битове трябва да се направят огледални и число, оказващо колко бита трябва да се използват за операцията. Създава времаенна променлива и я инициализира с 0, след което върти цикъл от 1 до броя на използваните битове. На всяка итерация премества наляво разрядната решетка на временната променлива, след което прави побитово ИЛИ на временнтата променлива с с побитово И от предаденото число и 1. Най-накрая премества разрядната решетка на числото с 1 бит надясно. Резултататъ след всички итерации е стойността на огледалното побитово число.

Пример: mirnum2(4,3)

Num = 4 🡪 [1 0 0]

Bits = 3

Tmp = 0 🡪 [0 0 0];

k = 1

Tmp = bitshift(Tmp,1) 🡪 [0 0 0] = 0

Val = bitand(4,1) 🡪 [1 0 0]\*[0 0 1] = [0 0 0]=0

Tmp = bitor(Tmp,Val) 🡪 [0 0 0]+[0 0 0] = [0 0 0]=0

Num = bitshift(Num,-1) 🡪 [0 1 0] = 2

k = 2

Tmp = bitshift(Tmp,1) 🡪 [0 0 0] = 0

Val = bitand(Num,1) 🡪 [0 1 0]\*[0 0 1] = [0 0 0]=0

Tmp = bitor(Tmp,Val) 🡪 [0 0 0]+[0 0 0] = [0 0 0]=0

Num = bitshift(Num,-1) 🡪 [0 0 1] = 1

k = 3

Tmp = bitshift(Tmp,1) 🡪 [0 0 0] = 0

Val = bitand(Num,1) 🡪 [0 0 1]\*[0 0 1] = [0 0 1]=1

Tmp = bitor(Tmp,Val) 🡪 [0 0 0]+[0 0 1] = [0 0 1]=1

Num = bitshift(Num,-1) 🡪 [0 0 0] = 0

Край на цикъла  
 Out = Tmp = 1

slashstringn – Функция за обработка и намиране на наклонена чера в стринг, която взима два параметъра като стринг и число. Ако числото е нула, функцията връща всички символи след последната наклонена черта до края, ако е положително, връща брой зададени символи от последната наклонена черта до края, ако е отрицателно връща позицията на последната наклонена черта в стринга.

Пример:

slashstringn(asd\asd,2) 🡪 ‘as’

slashstringn(asd\asd,0) 🡪 4

slashstringn(asd\asd,(3 или повече)) 🡪 “asd”

slashstringn(asd\asd,(-1 или по-малко)) 🡪 ‘asd\’

mirrornum – като mirnum2, но с използване на int2bits, bits2int и swaparr – работи като функция, на която се предават същите параметри като при mirnum2, с тази разлика, че операциите се извършват над масиви от битове

isMp3Format(path) – валидира MP3 файл, като изполва пътя към него

и хващане на грешка при четене

Пример: isMp3Format(‘D:\test.txt’) ans = 0

isMp3Format(‘D:\test.mp3’) ans = 1

fixInputin2Col() – преобразува матрица от nxk в nx2

Пример: fixInputin2Col([1,2,3;4,5,6]) 🡪 [1,2;4,5])

fixInputin2Col([1;4]) 🡪 [1,1;4,4])

Пример: mirrornum (4,3)

in = 4 🡪 [1 0 0]

l = 3

bits=int2bits(in,l) = [1 0 0]

bits=swaparr(bits,1) = [1 0 0] 🡪 [0 0 1]

out=bits2int(bits) = 0\*2^2+0\*2^1+1\*2^0 = 1

Следват алгоритми за изчисление на бързото преобразование на

Фурие в средата на Matlab, чрез реализиране на исновната операция – наречена „пеперуда“. Инициялизацията на трансформацията на Фурие представлява на базата на даден входен масив да се снемат необходимите данни за неговото обработване. Като за начало масиват може да не съдържа елементи, чиито брой е точно степен на числото 2 ( 1,2,4,8,16 ... ), затова входния масив се допълва с необходим брой нули за да стане броя на неговите елеменит кратен на 2. Това се реализира с функцията “zerofil” спомената по-горе, след което се взима броя на точките с врадената функция “length” и се заделя памет за операционните масиви ка.то например А – за стойностите на DFT между отделните фази и W – за необходимите фазови фактори. Хубаво е да се изчисли и броят на половината от точките на DFT, понеже това довежда до по-малко умножения при изчисляване на комплексния фазов фактор, а ако не разполагаме с необходимия хардуер, това довежда до видимо намаляване на производителността на изчисленията.

myfft1 – трансформация на Фурие с използване на алгоритъм, който разделя каскадно преобразуванието на Фурие на две половини, като на всяка итерация броя на изчисленията се намалява логаритъм при основа 2 пъти. Стойностите на Wn се смятат само веднъж, с цел по-голямо бързодействие.

Пример: myfft1([1 2 3 4]’)

Векторно-матричен алгоритъм input = [1 2 3 4]

sz=size(input); размерността на входния сигнал

if(sz(1) > sz(2))

input = input'; myfft1 смята DFT като ред (трансп. при необх)

Tflag = 1; запазваме състояние на транспонирания сигнал

else

Tflag = 0; ако не е нужно не го запазваме

end

r=log2(max(sz)); r = 2 фази

p=ceil(r); p = 2 пеперуди / фаза

Y=zerofil(input); допълване с нули

N = 2^p; N = 4 брой точки

N2=N/2; N2 = 2 брой точки / 2

YY = -pi\*sqrt(-1)/N2; YY = 0 - 1.5708i единична стойност на W

WW = exp(YY); WW = -1i основна степен за N = 4

JJ = 0 : N2-1; JJ = [0 1] степени на W

W=WW.^JJ; W = [1, - 1i] масив от стойностоте на W

p – 1 = 1 цикъла се извърта един път

u=Y(:,1:N2); u = [1 2] първа половина от input

v=Y(:,N2+1:N); v = [3 4] втора половина от input

t=u+v; t = [4 6] горната част на пеперудата

S=W.\*(u-v); S = [-2 2i] долната част на пеперуда

Y=[t ; S];

U=W(:,1:2:N2); U = 1

W=[U ;U];

N=N2; N = 2

N2=N2/2; N2 = 1

u=Y(:,1);

v=Y(:,2);

T=[u+v;u-v];

if(Tflag) гледаме дали сигнала трябва да се транспонира

Y = T;

else

Y = T';

end

Y =

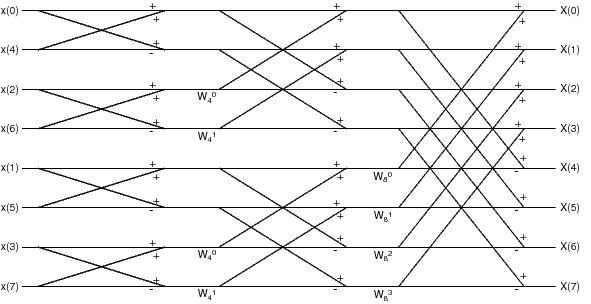
10.0000

-2.0000 + 2.0000i

-2.0000

-2.0000 - 2.0000i

myfft2 – трансформация на Фурие с използване на алгоритъма от учебника по ОДС – или извесният ни Cooley-Tukey алгоритъм. Той може да бъде обяснен със следния граф, където х(k) са измерените и пренаредени с функцията mirrnum2 стохности на сигнала в областта на времето, a X(k) е масив от комплексни числа, модолите на всяко от които съответства на стойностите на разложението на сигнала x(k) в областта на честотите. Тук смятането на Wn се извършва само един път, като стойностите се записват в масив, а взимането на съответното Wn става чрез обръщение по индекс о масива.



Функцията не се изпълнява толковова бързо колкото myfft1 и myfft3, но за сметка на това алгоритъма е много прегледен и подреден, като връща и отчет за направените пеперуди за всяка фаза, както и степените на Wn. От начало инициализираме, след което заделяме памет за променливата “c”, съдържаща състоянията във всяка фаза, след което приавим операцията bit-reversed чрез използване на функцията mirrnum2 и запулваме празня масив със стойности на Wn. Винаги започваме от първата фаза, затова дигаме най-младшия бит в стойност 1-ца и извъртаме цикъл по “k” от 1 до броя на точките, в който се смята текущата стемен на фазовия фактор с функцията bitand, след което една “пеперуда” се реализира чрез двойно преминаване през условието. Алгоритъма продължава нататък с изваждане на отчет за групиранията на елементите, степента на Wn и флаг, който ни показва през коя част от условието е минал интерпретатора. Най накрая прехвърля стойностите от масива за изчисление на следващата фаза, зададена от следващия бит, и така докато не свършат фазите за обработка.

Пример :   
[y,s] = myfft2([1 2 3 4]')

y = 10.0000

-2.0000 - 2.0000i

-2.0000

-2.0000 + 2.0000i

s = 1 1 2 0 1

1 2 1 0 -1

1 3 4 0 1

1 4 3 0 -1

2 1 3 0 1

2 2 4 1 1

2 3 1 0 -1

2 4 2 1 -1

Променливата “s” е с 8 реда, което означава, че са направени 4 “пеперуди” като цяло. Първата колона съдържа управляващия бит за съответната фаза ( 1 🡪 20 първа фаза, 2 🡪 21 втора фаза ....) , която Matlab вътрешно изчислява побитово, но в резутат от функцията bitshift виждаме само числото. Втората и третата колона показват групирането на елементите между входния и изходния масив за всяка фаза ( 1-ва фаза – 1-ви с 2-ри и обратно, 2-ра фаза – 1-ви с 3-ти и обратно ... ) . Четвъртата колона могат да се видят степените на фазовия фактор Wn. Както виждаме Wn = 1 за първа фаза. От петата колона се виждат реализираните събирания и изваждания за една “пеперуда”. Като цяло “s” може да се използва за изваждане на статистика за самия метод и неовото по-пълно разбиране.

myfft3 – трансформация на Фурие с използване на рекурсивен алгиритъм и векторно WN  чрез функцията Wn. Алгоритъма разделя сигнала на четна и нечетна последователност, като рекурсивната финкция се извиква в нерекурсивната, така преобразованието на Фурие се рзбива на вектори с четни и нечетни индекси, докато резултата се изроди в преобразувание на фурие от една точка.

Пример: myfft3([1 2 3 4]’)

sz=size(f);

if(sz(1) > sz(2))

f = f';

Tflag = 1;

else

Tflag = 0;

end

n = length(f); n = 4 брой точки

f = zerofil(f); допълване с нули

X = fftrec(f); извикване на рекурсивната програма

Първо влизане: ( Не влиза в проверката по дължина на масива )

n = length(f); n = 4 дължина допълнен масива

fe = f(1:2:n); fe = [1 3] нечетни индекси

fo = f(2:2:n); fo = [2 4] четни индекси

X1 = fftrec(fe);

Второ влизане с fe = [1 3]

n = length(f); n = 2 дължина допълнен масива

fe = f(1:2:n); fe = [1] нечетни индекси

fo = f(2:2:n); fo = [3] четни индекси

X1 = fftrec(fe); X1 = 1

X2 = fftrec(fo).\*Wn(n); X2 = 3 \* 1 = 3

F1 = X1 + X2; F1 = 4

F2 = X1 - X2; F2 = 1 – 3 = -2

F = [F1 F2]; F = [4 -2]

Второ влизане край

X2 = fftrec(fo).\*Wn(n);

Трето влизане

n = length(f); n = 2 дължина допълнен масива

fe = f(1:2:n); fe = [2] нечетни индекси

fo = f(2:2:n); fo = [4] четни индекси

X1 = fftrec(fe); X1 = 2

X2 = fftrec(fo).\*Wn(n); X2 = 4

F1 = X1 + X2; F1 = 2 + 4 = 6

F2 = X1 - X2; F2 = 2 - 4 = -2

F = [F1 F2]; F = [6 -2]

Трето влизане край

X1 = [4 -2]  
 X2 = [6 -2].\*[1 –i] = [6 2i]

F1 = [4+6 -2+2i]

F2 = [4-6 -2-2i]

F = [F1 F2];

Иизлизане от рекурсията и довършване, след което следва проверка на флага за транспониране и реверсиране ако се наложи (fftrec смята DFT използвайки сигнали - редове)

T = X(1:n);

if(Tflag)

X = T';

else

X = T;

end

end

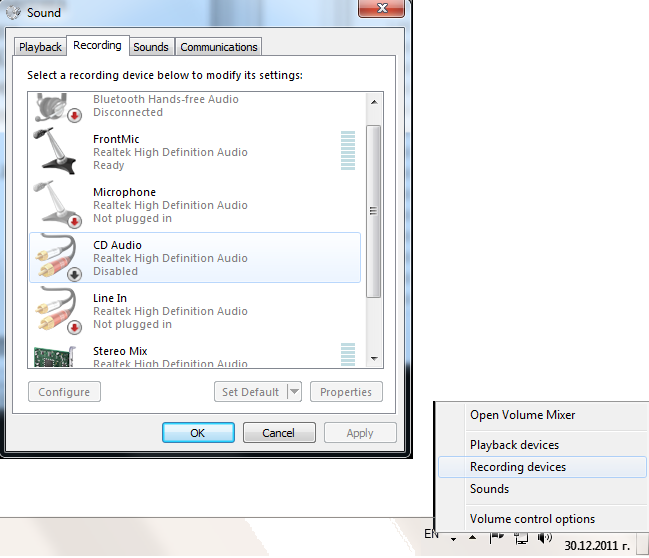
X = 10.0000

-2.0000 + 2.0000i

-2.0000

-2.0000 - 2.0000i

recsig – записва сигал, докато се възпроизвежда – работи като функция, на която се предават параметри колко време да запизва от засадено от потребителя “recording” устойство. Видовете устройства могат да се видят на фигурата

  
Функцията връща времето и записания сигнал ( най-често двуканален ). Първоначално се започва с инициализация, където се задават параметрите на записа – честота на дискретизация, брой битове, канали. След това се влиза в безкраен цикъл, който изважда входните устройства на потребителя и го кара да избере. Ако усройството е валидно се продължава надолу до създаване на обект от тип “audiorecorder”.

>> rec

BitsPerSample: 16

BufferLength: 0.1000

CurrentSample: 1

DeviceID: 0

NumberOfBuffers: 10

NumberOfChannels: 2

Running: 'off'

SampleRate: 44100

StartFcn: []

StopFcn: []

Tag: 'Sampled signal'

TimerFcn: []

TimerPeriod: 0.0500

TotalSamples: 220500

Type: 'audiorecorder'

UserData: 0

Структората има и други полета: BitsPerSample – показва колко голяма разрядна решетка, която се използва за съхранение на всяка стойност от сигнала; BufferLength – показва колко време от сигала е записано в един буфер; NumberOfBuffers – показва колко буфера съдържа обекта; CurrentSample – поле, което е само за четене и показва до каде е стигнала възпроизвеждането до момента; DeviceID – показва кое входно устройство ще се използва при записа на сигнала; NumberOfChannels – брой независими канали на аудио сигнала; Running – флаг дали се възпроизвежда или не; SampleRate – честота на дискретизация; StartFcn – функция, която трябва да се изпълни при стартиране на аудиото; StopFcn – функция, която трябва да се изпълни при спиране на аудиото; Tag – стринг заданен от потребителя, който представлява някакъв стор информасия за дасения сигнал; TimerFcn – функция, която трябва да се изпълни на период от време, зададен от TimerPeriod в секунди; TotalSamples – брой на дискретните нива на сигнала в целия файл; Type – тип на обекта ( в случая “audiorecorder” защото заисваме сигнал ); UserData – даден вид данни ( числови или стринг за четене или запис ) зададени от потребителя.

и празен масив, в който ще се съхрани записания сигнал. Влиза се в цикъл, който се управлява от оставащото време за записване, което се съхранява в полето на обекта rec.UserData. Ако времето за запис е по-малко от 40 секунди, сигнала се записва изцяло, но ако е повече, функцията recordblocking неможе да поеме по-големи времена, и тогава се налага използването на getaudiodata. Чрез нея може да се запише една секунда от входния сигнал, след което да се долепи към масива за изход. Потребителския интерфейс на всяка итерация изписва колко % от времето остава да се записва, използвайки функцията percent\_q.

Пример:

>> [t,y]=recsig(0,0,1);

Dev >Microphone (VF0530 Live! Cam Ch< Ver>131.20< ID >0

Dev >FrontMic (Realtek High Definiti< Ver>6.1< ID >1

Dev >Stereo Mix (Realtek High Defini< Ver>6.1< ID >2

Dev >CD Audio (Realtek High Definiti< Ver>6.1< ID >3

Device ID? >>0

100% Complete

>>

openmp3 – аудиплеър за средата на MATLAB с вградена едномерна трансформация на Фурие. Алгоритъмът е следния: В началото използва функцията mp3read, която споменахме в началото. Към нея се предава пътя до файла заедно с името като стринг, след което тя връща сигнала прочетен от файла, честотата на дискретизация, разрядността на всеки сампъл и допълнителна информаця – например метода на комресия, брой каналите изпълнител и т.н.

След това се използва променливата “audiodevinfo”, в която се съсържа информацията за всички входно-изходни аудио устройства

в Matlab:

>> audiodevinfo

ans =

input: [1x4 struct]

output: [1x2 struct]

Извърта се цикъл който има за цел да обходи елементите и да събере информация за тях, като например име на устройството, версия на драйвера и най-вече ID, което по-нататък се използва за създаване на обект от тип “audioplayer”, като се записва в полето “DeviceID”.

>> g = audioplayer(y,44100,16)

BitsPerSample: 16

CurrentSample: 1

DeviceID: -1 (по подразбиране от Windows7)

NumberOfChannels: 2

Running: 'off'

SampleRate: 44100

StartFcn: []

StopFcn: []

Tag: ''

TimerFcn: []

TimerPeriod: 0.0500

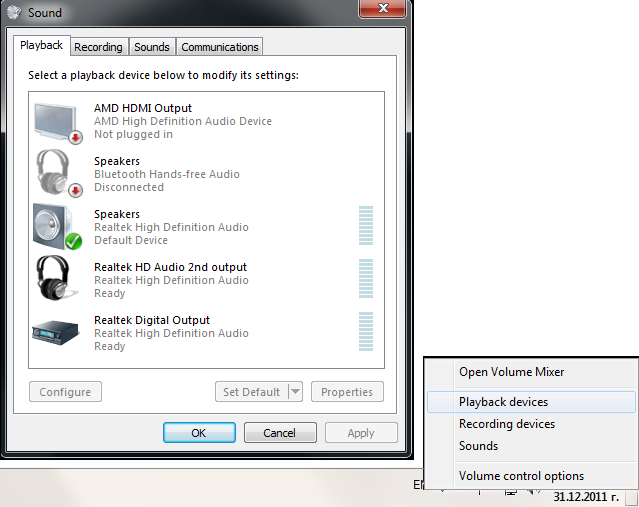
TotalSamples: 220500

Type: 'audioplayer'

UserData: []

Тук полетата са подобни на тези при запис, с тази разлика, че няма полета отговарящи за буферите и типа е “audiopleyer”

Завърта се още един цикъл, който има за цел да отработи потребителския интерфейс, като даде избор на потребителя да избере дадено изходно устройство, което да се използва за възпроизвеждането, имайки на предвид, че броят на устройствата е крайна целочислена стойност. Те са дадени на следната фигура:



Цикълът приключва, когато даденото устройство е валидно и неговото ID отговаря на интервала (0;N). Алгоритъма продължава с изчисление на продължителността на аудиото, като преди това запазва честотата на дискретизация на страни. Следва изобразяване на информацията свързана с файла, като най накрая в инициализацията се изтриват всички временни променливи, кото не са необходими по нататъп за изчисленията. След това процесора влиза в безкраен цикъл, в който чака команада от потребителя. Командите могат да бъдат на-различни:

p – паузира възпроизвеждането чрез използване на командата, към която се предава audio-обекта pause(mp3)

r – продължава възпроизвеждането чрез използване на командата, към която се предава audio-обекта resume(mp3)

s – спира възпроизвеждането чрез използване на командата, към която се предава audio-обекта stop(mp3)

q – спира възпроизвеждането чрез използване на командата, към която се предава audio-обекта stop(mp3), с таз разлика, че изчиства всички променливи и излиза от програмата.

o – отваря желан от потребителя файл, чрез рекурсивно извикване в самата себе си, като му предава пътя. Подразбира се че файла трябва да е валиден.

seek – прескача на даден процент от файла и започва да възпроизвежда оттам 🡪 работи, като използва функцията percent\_q

за да зичисли каква част е текущия указател от всички стойности в проценти и го показва, за да може да се разбере до колко % е стигнало възпроизвеждането, след което потребителя преценява това и задава процент, който в последствие се конвертира до цяло, положително число чрез използване на функцията part\_q и възпроизвежда оттам чрез командата play, към която се предава масив от индекси

rate – променя честотата на дискретизация на audio-обекта, като потребителя задава стринг, който след това се конвертира в число чрез функцията str2double и получената стойност се записва в полето на обекта отговарящо за честотата на дискретизация.

dir – показва списък от фаилове в директориятa на песента

info – показва информация за обекта

clc – изчиства екрана

plist – настройка на плейлиста на плейъра чрес прогами Add Rem и Sel

v 🡪 p – прави графика на сигнала и преобразованието на Фурие в реално време. В началото се прави графическа инициализация – например колко голям буфер да се направи, колко е разрешаващата способност по честота, броя точки за DFT и разбира се вида на прозореца ( по подразбиране “blackman” ). След това се извърта безкраен цикъл, който спира или при край зададен от потребителя с натискане на бутон “s” или при достигане на края на файла в режим възпроизвеждане. След това разделяме канала на ляв и десен като взимаме даден брой стойности, които в последствие претегляме с избран прозорец. Смята се DFT с някоя от функциите

myfft1, myfft2, myfft3, като може да се зададе и вградената в Matlab-a и абсолютната стойност на комплексните числа, след което следва визуализирането. То се прави като се отваря фигура с функцията figure, където в нейния аргумент се задава предварително изчисления хендъл на следващата по номер фигура. Това се прави за да може потребителя едновременно да слуша музика, да наблюдава процеса на възпроизвеждане и да разглежда други отворени фигури, които е създал. Командата subplot нарежда графиките в матрица 4x4, като отгоре се визуализират левия и съответно десния канал на сигнала, претеглени с прозореца в зададени предварително граници на фигурата. Долния ред, отново с командата plot се визуализират DFT на левия и десния канал на претегления сигнал. Хендъла на фигурата винаги е по-голям с 1 от последната отворена фигура.

v 🡪 w,s,m – вградена едномерна трансформация на Фурие, където трансформациите от минали моменти се изобразяват от функцията waterfall ( mesh или surf ) с даден коефициент на забравяне. Рзличава се най-вече в алгоритъма на инициализация на визуализацията. Алгоритъма се изразява в следното: инициализация, заделяне на памет за “бягащата” DFT за левия и десния канал, влизане в безкрайния цикъл, взимане и претегляне на стойностите на аудио-сигнала с зададен прозорец ( по подразбиране blackman ), изчисляване на една колона от DFT на леваи и десния канал, преместване на стойностите с една колона надясно и на двата канала със shiftArr, обръщане на стойностите в колоната на DFT на двата канала с функцията swaparr за да може ниските хармоници да не скрият високире на графиката и записване в първата колона от буфера. Буферите са с константна дължина и се изобразяват от функцията waterfall като 3D повърхнина от дискретни линии на двата долни прозореца на subplot на зададената фигура, чиито хендъл е винаги с 1 по-голям от предишната отворена фигура.

Пример:

f = ‘ G:\MP3\test\Cascada-2009\07 Breathless.mp3 ‘

>> openmp3(f)

Dev >Speakers (Realtek High Definiti< Ver>6.1< ID >0

Dev >Realtek Digital Output (Realtek< Ver>6.1< ID >1

Dev >Realtek HD Audio 2nd output (Re< Ver>6.1< ID >2

Device ID? >>0

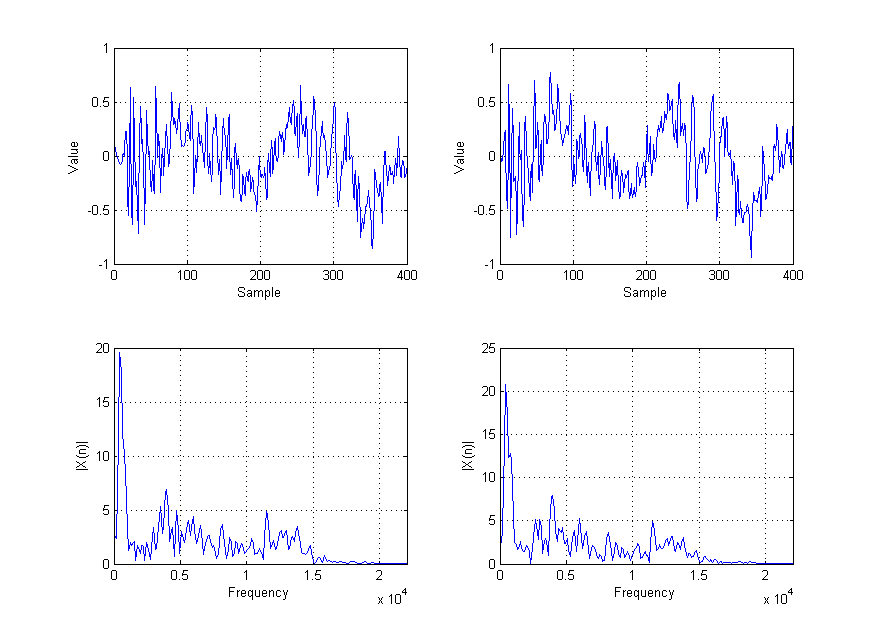
Now playing >> 07 Breathless.mp3

Audio length >3.1789< mins >2< chanels

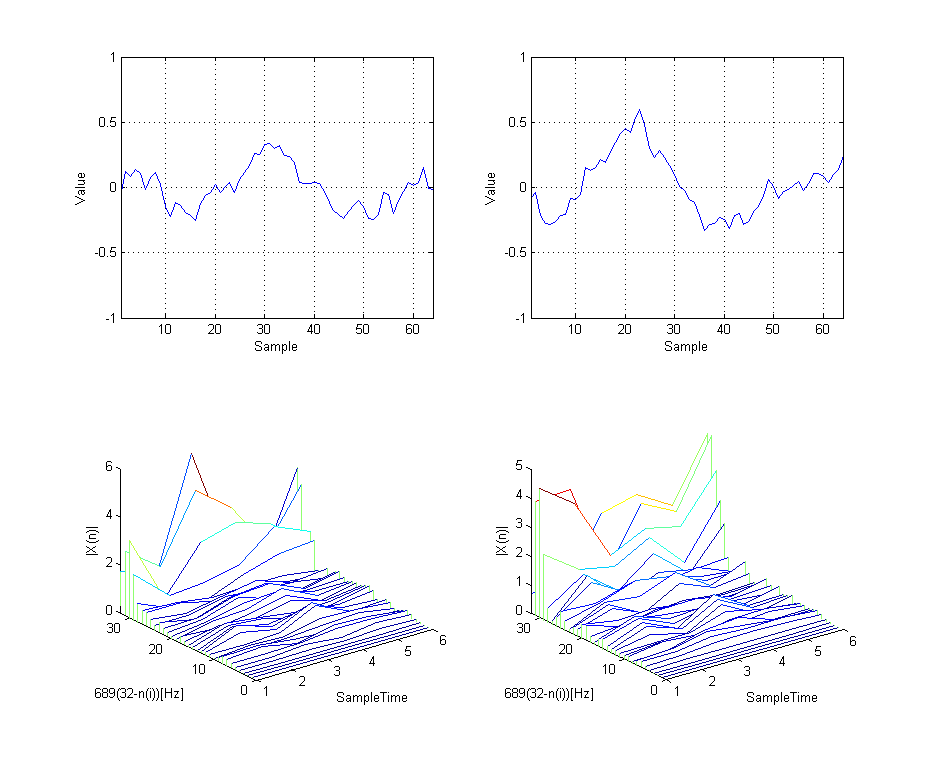
Mpeg version >1< Layer >III

Command >> v

Command >> View >> p

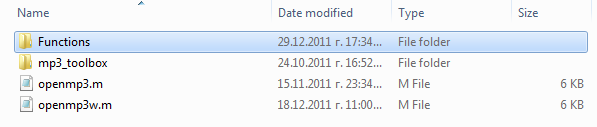


Command >> View >> w ( w waterfall, m mesh, s surf )

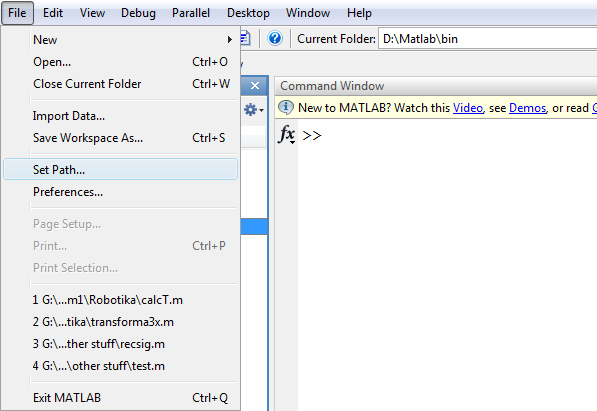


1. Сваляне на билблиотеките от [тук](http://labrosa.ee.columbia.edu/matlab/), като всеки файл отиде на определеното за него място – както следва в mp3\_toolbox\_v2.0  
   mp3\_toolbox\mp3reader 🡪 mp3info, mpg123, mp3read  
   mp3\_toolbox\mp3write 🡪 lame, lame\_enc, mp3write

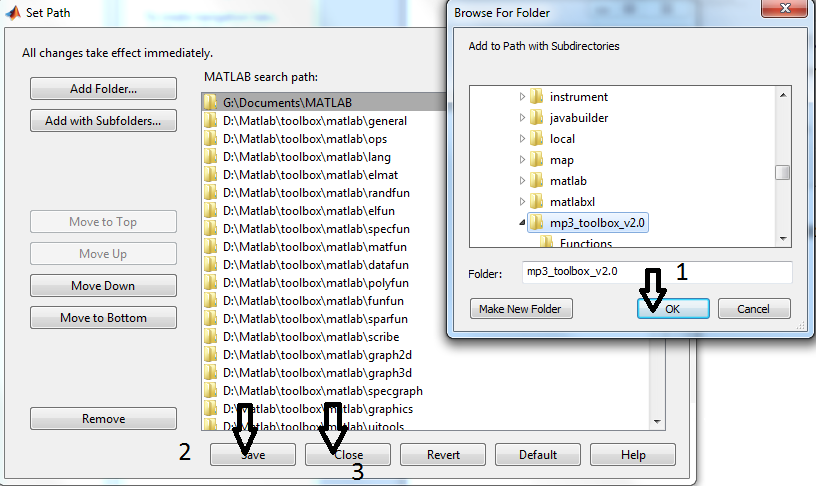
Създава се папка на име “mp3\_toolbox\_v2.0\Functions”, където се пеместват всички финкции от точка 3., необходими на пеъра за да роботи, с изключение на “openmp3” и “openmp3w”.След успешно завършената подредба, директорията „\toolbox\mp3\_toolbox\_v2.0“ трябва да изглежда по този начин.



След това се пуска Matlab-a и се отива на File 🡪 Set Path



Там се отваря следния прозорец, в който се задава пътя до библиотеката, след което ok 🡪 save 🡪 close



1. Изводи

Като цяло разработената библиотека дава един мощен потребителски апарат за управление на сигнали ( четене, запис и анализ ) в реално време, както и добра визуализация. Съдържа множество функции, за операции с матрици, полезни за много други потребителски програми освен дискретното преобразувание на Фурие, описано тук.

Бяха разгледани три отделни функции за изчисление на DFT в реално време като всяка една може да приема едномерен сигнал с произволна размерност, като ред или колона. Нейното по-нататъшно доразвиване би могло да бъде по потребителския интерфейс, включващо плейлист и еквалайзер.