Promjena frekvencije audio signala primjenom Fourierove transformacije

Dželajlija, Danijel

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:670689

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski studij

PROMJENA FREKVENCIJE AUDIO SIGNALA PRIMJENOM FOURIEROVE TRANSFORMACJE

Diplomski rad

Danijel Dželajlija

Osijek, 2016.

Sadržaj

1. UVOD 1
1.1. Zadatak diplomskog rada 1
2. FOURIEROVA TRANSFORMACIJA 2
2.1. Objašnjenje Fourierovog teorema 3
2.2. Broj sinusoida potreban za realizaciju određenog signala 4
2.3. Pripremanje recepta za dobivanje željenog oblika 6
2.4. Diskretna Fourierova transformacija 8
3. PRIMJENA KRATKE FOURIEROVE TRANSFORMACIJE KOD KODIRANJA 11
3.1. Problem razlučivosti frekvencije 11
3.2. Poveznica faze i frekvencije 13
3.3. Faktor preklapanja i njegov utjecaj na signal 16
3.4. Promjena frekvencije 20
3.5. Amplitudni i frekvencijski spektar 21
3.6. Zapisivanje u kod 22
3.7. Rezultat transformacije 26
4. C# I OSTALI KORIŠTENI ALATI 27
5. ZAKLJUČAK 30
LITERATURA 31
SAŽETAK 32
ABSTRACT 33
ŽIVOTOPIS 34

1. UVOD

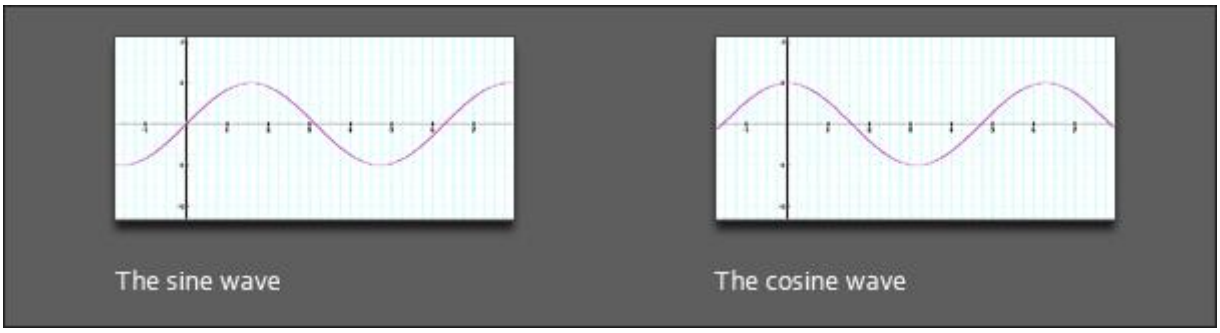
S obzirom na sve veću brzinu stolnih računala, sve veći broj teških računalnih operacija kao što su računanje Fourierove transformacije uzorkovanog audio signala postale su dostupne širokom pojasu korisnika. S obzirom da je to bio proces koji se tradicionalno implementirao na sustave DOS, ili na snažna računala koja su mogli imati samo neki korisnici, Fourierova transformacija se može računati na gotovo svim računalima prosječnih komponenti. Uvođenjem pojma frekvencije u prikaz signala, ovaj proces čini se prikladan za primjene od kojih su neke: promjena visine tona audio signala, ali uz to ne mijenjajući njegovu duljinu, ili promjena duljine tona, ali uz konstantnu visinu tona. Ova aplikacija ima značajnu praktičnu primjenu u današnjim sustavima za audio obradu. U ovom radu jasno će se objasniti proces promjene visine tona s obzirom na oblik signala u frekvencijskoj domeni pod pretpostavkom da je produljivanje ili skraćivanje vremena analogno. Promjena visine tona koristeći tonski koder radi se tako da se mijenja bazno vrijeme signala i tako da se koristi pretvaranje uzoraka na izlazu, kako bi se postigla promjena visine tona, ali uz zadržanu duljinu. Također u nekim primjenama se koristi resinteza oscilatorskog spremnika za promjenu visine tona, što je u biti neučinkovito. Ovdje se neće koristiti koder u njegovom poznatom obliku nego će se radije koristiti sličan proces u kojem će se direktno promijeniti visina tona ali uz zadržanu duljinu. Proces koji će se koristiti ispod koristiti će Fourierovu i inverznu Fourierovu transformaciju kako bi se ugradila promjena visine tona i uključilo prikladno anti-preklapanje u frekvencijskom području.

1.1. Zadatak diplomskog rada

Zadatak ove teme je opisati Fourierovu transformaciju i načine njene primjene u modulaciji ulaznog audio signala. U praktičnom dijelu zadatka potrebno je napraviti real-time snimanje, promjenu frekvencije te vizualizaciju ulaznog audio signala snimljenog putem glazbenog instrumenta.
Tehnologije: C/C++, MATLAB

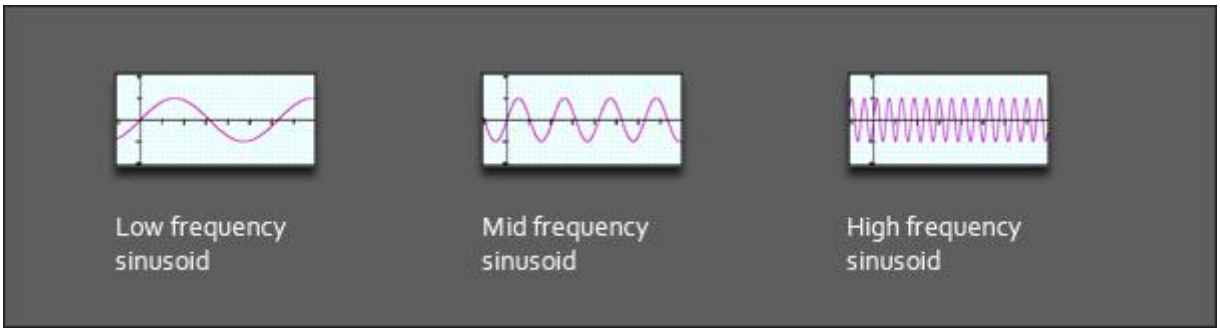
2. FOURIEROVA TRANSFORMACIJA

Za početak razumijevanja Fourierove transformacije neće trebati puno znanja. Potrebno je samo znati zbrajati, množiti te znati što je sinusni, kosinusni val te kako izgledaju.



Slika 2.1 sinusni i kosinusni val; Izvor: Stephan Bernsee's blog, 21.10.1999.

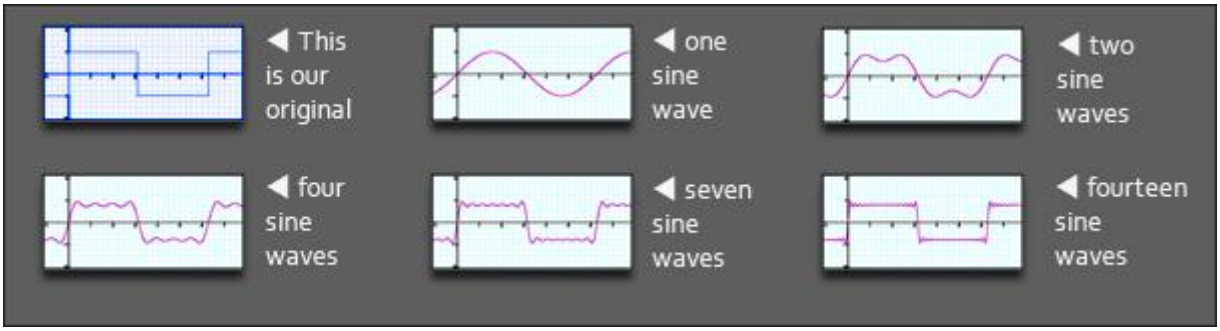
Sa slike se jasno vidi da su oba vala periodična, tj iako se u vremenu mijenjaju, nakon određenog perioda će poprimiti isti oblik. Jedna od očitih razlika je da kosinusni val počinje sa svog maksimuma a, a sinusni iz nule. Kada bi se sada promatrala oba vala kako se mijenjaju u vremenu, ne postoji osnova na temelju koje bi se moglo zaključiti koji je signal krenuo iz nule koji iz maksimuma. Pošto se ne može razlučiti sinusni od kosinusnog vala, u praksi se oba nazivaju sinusoida. Važno svojstvo sinusoide je frekvencija, koja govori koliko maksimuma i minimum ima u određenom vremenskom intervalu. Visoka frekvencija znači da u tom određenom vremenskom intervalu postoji puno maksimuma i minimuma, dok kod niske frekvencije nema.



Slika 2.2 Razne frekvencije sinusoida;Izvor: Stephan Bernsee's blog, 21.10.1999.

2.1. Objašnjenje Fourierovog teorema

Jean-Baptiste Joseph Fourier mnogo je doprinio svijetu svojim znanjem, ali jedna od najznačajnijih stvari bila je vezana za vođenje topline u materijalima. Otkrio je jednadžbu koja opisuje kako se toplina širi u određenom prostoru i, riješio ju je tako što je izračunao praktički beskonačnu seriju trigonometrijskih funkcija sinusa i kosinusa. Konkretno za slučaj koji je zanimljiv u ovome radu, ono što je Fourier otkrio bilo je to da se svaki signal, ma kako bio kompleksan, može prikazati kao suma funkcija sinusoida koje su međusobno pomiješane.

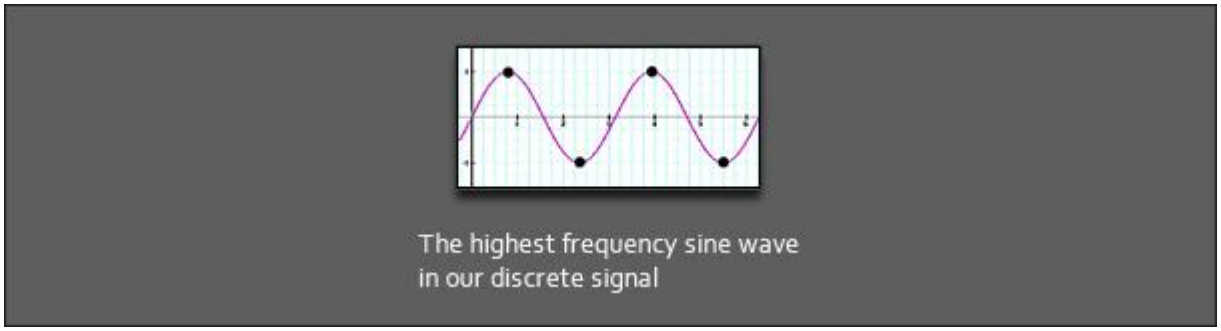


Slika 2.3 Dobivanje željenog signala dodavanjem sinusoida;Izvor: Stephan Bernsee's blog, 21.10.1999.

Na prvoj sličici ove slike vidi se prikazan signal, a ostale slike prikazuju kako se on dobije dodavanjem većeg broja sinusoida, tj. da je taj signal zapravo kombinacija određenog broja sinusoidnih funkcija (koje će se nazvati segmenti sinusoide) koje su pomiješane na određeni način, tj. po nekom "receptu". Poslije će biti rečeno nešto više o tom receptu. Kao što je vidljivo iz slike, što se više sinusoida koristi to će konačni rezultat biti sličniji početnom, željenom izgledu signala. U svakodnevnim situacijama signali su neprekidni (može ih se mjeriti u beskonačno malim intervalima preciznošću koja je ograničena jedino mjernim instrumentom) i bilo bi potrebno bezbroj sinusoida da se bilo koji signal prikaže. Tu se otkriva poanta digitalnog procesuiranja signala, gdje se koriste samo uzorci koji se mjere u točno određenim vremenskim intervalima, ali imaju ograničenu preciznost. Iz toga razloga, za prikaz nekog signala u svakodnevici, nije potrebno beskonačno mnogo sinusoida, ali i dalje je potreban velik broj. Poslije će se reći nešto o tome koliko zapravo iznosi taj veliki broj sinusoida. Za sada je najvažnije da se zna da se svaki signal, koliko god bio kompliciran, može prikazati uz određen broj sinusoida, ali naravno uz odgovarajući "recept".

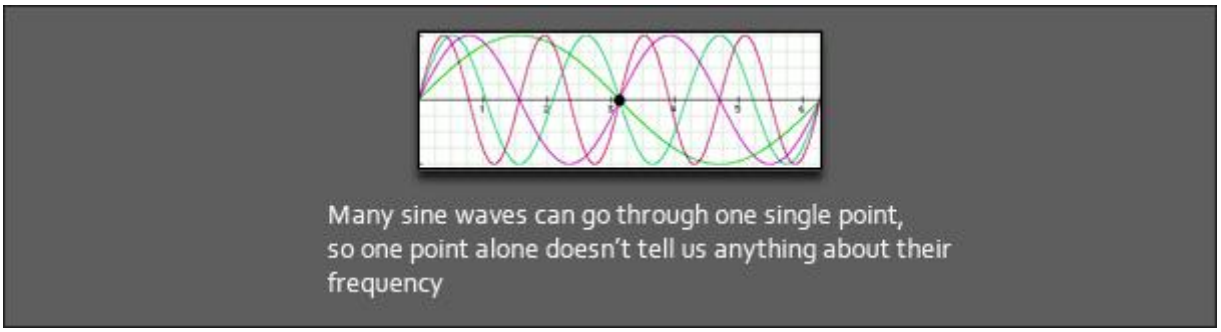
2.2. Broj sinusoida potreban za realizaciju određenog signala

Pitanje koje se postavlja je koliko je sinusoida potrebno kako bi se dobio neki signal iz računala. U većini slučajeva radi se sa signalima svakodnevice, koji mogu imati iznimno kompliciranu strukturu, tako da se ne može znati unaprijed od koliko se sinusoida sastoji određeni val. Ovome se može pristupiti intuitivno: neka se pretpostavi da se uzme tisuću uzoraka signala. Sinusni val sa najkraćom periodom koji može biti prisutan ima maksimume i minimum koje međusobno izmjenjuju za svaki uzorak. Dakle, sinusni val sa najvećom frekvencijom ima 500 maksimuma i 500 minimuma u tih 1000 uzoraka, dakle svaki drugi uzorak je zapravo maksimum. Crne točkice na dolje prikazanoj slici označavaju uzorke, tako da sinusni val sa najvećom frekvencijom izgleda ovako:



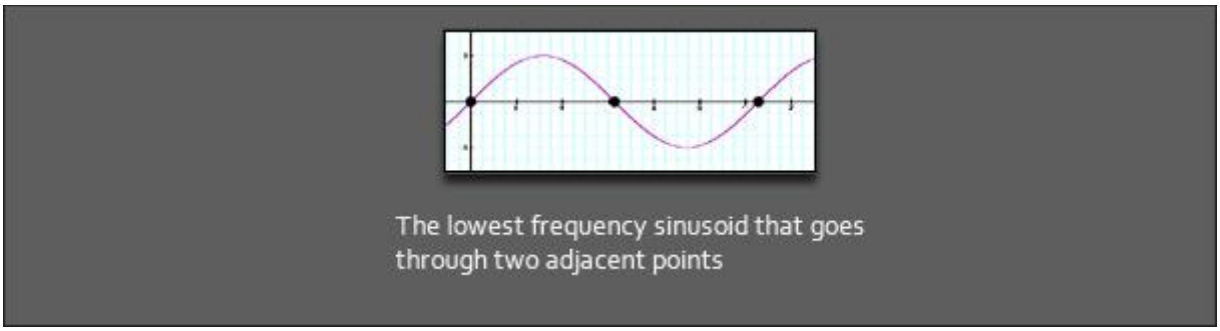
Slika 2.4 Najviša frekvencija sinusnog vala kod diskretnog signala; Izvor: Stephan Bernsee's blog, 21.10.1999.

Sada treba vidjeti kako izgleda sinusoida sa najmanjom frekvencijom. Ako je tu samo jedna točka uzorka, treba se zapitati slijedeće: Kako će se izmjeriti maksimume i minimume sinusnog vala koji prolazi kroz ovu točku? Nemoguće je, jer postoje mnoge sinusoide različitih perioda koje prolaze kroz tu točku, tako da se ne može otkriti kolika je frekvencija tog signal.



Slika 2.5 Prolazak raznih sinusoida kroz jednu točku; Izvor: Stephan Bernsee's blog, 21.10.1999.

Naprotiv, ako su dana dva uzorka moguće je odrediti. To se vidi na sljedećoj slici:



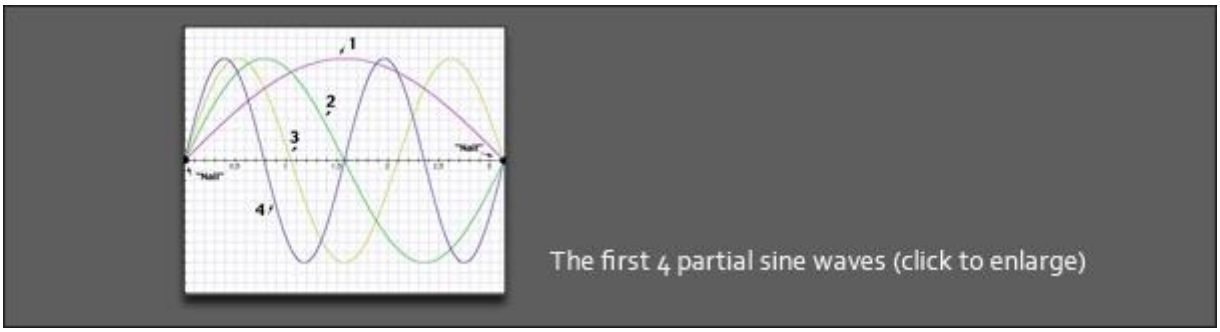
Slika 2.6 Najniža frekvencija koja prolazi kroz dvije susjedne točke; Izvor: Stephan Bernsee's blog, 21.10.1999.

Sada se može zamisliti dvije krajnje točke kao dva čavla između kojih je napeta žica (na slici iznad vide se tri točke, ali je zapravo potrebno imati samo dvije lijeve točke kako bi se odredila frekvencija). Najmanja frekvencija koju se može vidjeti je gibanje žice (napete između čavala) naprijed – nazad, kao što to radi sinusoida na slici. Kada bi bilo tisuću uzoraka, dva "čavla" koja bi se izabrala bili bi prvi i zadnji, tj. prvi i tisućiti uzorak. Iz iskustva sa žičanim glazbenim instrumentima zna se da se produljivanjem duljine žice (otpuštanjem žice na mašinici) frekvencija smanjuje. Dakle, može se očekivati da će se frekvencija - ako se udalje ova dva "čavla" još više – smanjiti. Ako se izabere dvije tisuće uzoraka, najniža frekvencija će sada biti još niža, jer su sada ti "čavli" postavljeni na prvo i na dvije tisućito mjesto. Konkretno govoreći, frekvencija će biti dvostruko manja jer su sada "čavli" dvostruko više udaljeni. Dakle što više postoji uzoraka to će najniža frekvencija biti manja. Ovo je važno kako bi se razumjelo nadolazeća objašnjenja.

Iz Slike 2.6 se može jasno vidjeti da će signal nakon dva uzorka ponovno početi rasti (putanja signala kod prvog i trećeg uzorka su jednake). To znači da dva susjedna uzorka obuhvaćaju pola sinusoide, tj. ili jedan pozitivni dio ili jedan negativni dio sinusoide.

Rezimirajući ono što je do sad rečeno, vidi se da se "bregovi" i "doline" viših frekvencija izmjenjuju svakim uzorkom, dok kod nižih frekvencija pola periode obuhvaća sve promatrane uzorke. Važan zaključak koji se može donijeti na temelju toga je taj da povećanjem broja uzoraka najniža frekvencija pada, dok najviša frekvencija ostaje ista, jer se ne mijenja razmak između uzoraka (što je ključna stavka kod određivanja najviše frekvencije) nego se mijenja razmak između prvog i zadnjeg uzorka, a pošto se najniža frekvencija određuje na temelju toga – ona se mijenja. Rezultat toga je da će biti potrebno više sinusoida ako se želi sastaviti duže signale nepoznatog sadržaja, jer se počinje od najniže frekvencije.

Za sada još uvijek nije određeno koliko je u konačnici potrebno sinusoida. Utvrđeno je kako odrediti najvišu i najnižu frekvenciju bilo kojeg segmenta ulaznog signala, a sad treba vidjeti koliko sinusoida ima između. S obzirom da je sinusoida najniže frekvencije razastrta između krajnjeg lijevog i krajnjeg desnog uzorka, podrazumijeva se da će i ostale sinusoide proći kroz prvi i zadnji uzorak, a primjer za to opet može biti gitara, kod koje je žica napeta između dvije točke i jasno je da će svaki val morati proći kroz obje te točke. To vodi do očitog zaključka da samo pola perioda prve sinusoide stane između dva uzorka, cijeli period druge, period i pola od treće, itd. , što se jasno vidi na slici:



Slika 2.7 Četiri parcijalne frekvencije u jednom periodu; Izvor: Stephan Bernsee's blog, 21.10.1999.

Ako bi se sada brojalo koliko segmenata ulaznog signala stane na ovaj način, došlo bi se do zaključka da je potrebno točno tisuću sinusoida, tj. onoliko koliko ima uzoraka.

2.3. Pripremanje recepta za dobivanje željenog oblika

Prethodno je bilo rečeno da se bilo koji dani signal može dobiti kao kombinacija sinusoida, ali nije bilo rečeno kako kombinirati sinusoide kako bi dobili određeni oblik. Ukoliko se pomoću sinusoida želi napraviti određeni signal, mora se izmjeriti još jedna veličina, a to je amplituda, tj. koliko je od svake sinusoide potrebno za sastavljanje ulaznog signala. Amplituda je udaljenost maksimuma od nule. Što je viša amplituda signala koji se sluša to će taj signal biti glasniji. Jedan primjer koji na neki način može povezati frekvenciju i amplitudu je sljedeći: ako se pri slušanju npr. glazbe jako čuje bas, tj. ako je on naglašen, zna se da veći udio imaju niske frekvencije. Dakle, generalno gledano, sinusoide niske frekvencije će imati višu amplitudu od sinusoida više frekvencije. Pri obavljanju analize biti će potrebno odrediti amplitudu svakog pojedinog segmenta sinusoide kako bi se odredio točan recept za željeni audio signal.

Sada je određeno koliko sinusoida je potrebno (broj ovisi u broju uzoraka koji su dani), da postoji gornja i donja granica frekvencije i da se na neki način mora odrediti amplituda svakog pojedinog segmenta sinusoide kako bi se dobio željeni oblik audio signala. No i dalje nije potpuno jasno kako dobiti točan recept. Neki intuitivan način bio bi da se amplitude sinusoida pokušaju naći uspoređivanjem sinusnog vala (čija je frekvencija poznata) sa izmjerenim uzorcima i na taj način saznati koliko su frekvencije slične. Ako su potpuno jednake zna se da sinusoida mora imati istu amplitudu, a ako se signal uopće ne poklapa sa referentnim, pretpostavit će se da ta frekvencija uopće nije prisutna. Treba se ipak zapitati kako bi se moglo usporediti poznati sinusni val i uzorkovani signal? Inženjeri koji se bave digitalnom obradom signala (DOS) pronašli su rješenje za to. Uzme se referentni sinusni oblik poznate frekvencije i jedinične amplitude (amplituda iznosi 1, kao što je to slučaj kod npr. sin90˚) i to se pomnoži sa uzorcima signala. Nakon zbrajanja tih pomnoženih rezultata, dobit će se amplituda određenog segmenta sinusoide.

Kako bi se ovo bolje ilustriralo, prikazat će se u kodu na slici ispod:

```
1  //
2  // Listing 1.1: The direct realization of the Discrete Sine Transform (DST):
3  //
4
5  #define M_PI 3.14159265358979323846
6
7  long bin,k;
8  double arg;
9  for (bin = 0; bin < transformLength; bin++) {
10
11     transformData[bin] = 0.;
12     for (k = 0; k < transformLength; k++) {
13
14         arg = (float)bin * M_PI *(float)k / (float)transformLength;
15         transformData[bin] += inputData[k] * sin(arg);
16
17     }
18
19 }
```

Slika 2.8 Direktna realizacija diskretne sinusne transformacije; Izvor: Stephan Bernsee's blog, 21.10.1999.

Ovaj primjer programa radi sljedeće: pretvara izmjerene uzorke pohranjene u *inputData[0…transformLength-1]*, u polje pripadajućih amplituda segmenata sinusoide - *transformData[0…transformLength-1]*. Prema općenitoj terminologiji, frekvencijske korake sinusoide naziva se bin-ovima. Njih se može zamisliti kao neke vrste spremnika u koje se sprema amplituda svakog pojedinog segmenta sinusoide koji se ispituje. Diskretna sinusna pretvorba je srodan proces kod kojeg se pretpostavlja da korisnik ne zna ništa o izgledu signala. U suprotnom, može se koristit druga, učinkovitija metoda za određivanje amplitude nekog segmenta sinusoide (ako

je unaprijed poznata frekvencija sinusnog vala, može se koristit Goertzel-ov algoritam). Postupak gore opisan u kodu se može obrnuti. Signal se jednostavnim zbrajanjem može (unutar granica numeričke preciznosti korisnika) savršeno rekonstruirati kada se znaju svi njegovi segmenti sinusoide. Ono što je gore u kodu napravljeno sa sinus funkcijom može se postići i kosinus funkcijom. Jednostavno se zamijeni *sin(arg)* sa *cos(arg)* kako bi se dobila izravna realizacija diskretne kosinusne pretvorbe. Kao što je rečeno na počeku rada, ne postoji način na koji se sinusoida snimljena u stvarnom okruženju može karakterizirati kao sinusni ili kosinusni val. Umjesto toga, uvijek se mjere sinusoide, tako da sinusna i kosinusna pretvorba nisu od prevelike pomoći u praksi, osim u nekim posebnim slučajevima (npr. sažimanje slika gdje svaka slika može imati značajke koje su dobrim dijelom sinusna ili kosinusna funkcija, kao što su velika područja iste boje koja se dobro mogu prikazati kosinusnom funkcijom). Sinusoida je čak malo općenitiji pojam od sinusnog i kosinusnog vala u tom smislu da može započeti u nekom proizvoljnom položaju uz svoj vlastiti period. Još je prije spomenuto da sinus kreće od nule, a kosinus od jedan. Ako se uzme sinusni val kao referenca, može se reći da je kosinusni val jednak sinusnom valu koji započinje četvrtinu periode kasnije (jer je maksimum, tj. 1 na četvrtini periode). Ovo odstupanje mjeri se u stupnjevima ili radijanima. Cijela perioda ima 360 stupnjeva, dakle može se reći da je kosinusni val sinusoida pomaknuta za 90 stupnjeva u odnosu na sinusni val. U velikom broju slučajeva se promatra neki signal nad kojim osoba koja ga mjeri nema kontrolu, pa tako ne može utjecati ni na to da taj val krene sa faznim pomakom jednakim nuli ili jednakim devedeset stupnjeva i zato je tako važno odrediti frekvenciju, fazu i amplitudu kako se bi taj val mogao jednoznačno odrediti u bilo kojem vremenskom trenutku. Uz sinusnu ili kosinusnu transformaciju korisnik je ograničen na fazni pomak od nula ili devedeset stupnjeva. Zbog svake sinusoide koja ima proizvoljan fazni pomak, njoj susjedne frekvencije će pokazivati lažne maksimume (pokušavaju “pomoći” analizi da na bilo koji način umetne signal tako da mu faza bude nula ili devedeset stupnjeva). Dakle ono što je potrebno je dobiti transformaciju koja će moći rješavati slučajeve kada sinusoida ima proizvoljnu fazu.

2.4. Diskretna Fourierova transformacija

Kod Fourierove transformacije za mjerenje frekvencija nekog signala koristi se sinusni i kosinusni val. To znači da se za svaku frekvenciju koja se trenutno promatra uspoređuje mjereni signal sa sinusnim i kosinusnim valom iste frekvencije. Ako mjereni signal izgleda više kao sinusni val, to znači da će sinusni dio transformacije imati veliku amplitudu. Isto vrijedi ako mjereni signal više naliči kosinusom valu. Ako mjereni signal izgleda kao obrnuti sinusni signal, tj. ako je ekstrem funkcije odmah poslije nule minus jedan, sinusni dio će imati veliku negativnu amplitudu, može se

pokazati da pozitivni i negativni predznak zajedno sa sinusnom i kosinusnom fazom mogu predstavljati bilo koju sinusoidu na danoj frekvenciji.

```
//
// Listing 1.2: The direct realization of the Discrete Fourier Transform***:
//

#define M_PI 3.14159265358979323846

long bin, k;
double arg, sign = -1.; /* sign = -1 -> FFT, 1 -> iFFT */

for (bin = 0; bin <= transformLength/2; bin++) {

    cosPart[bin] = (sinPart[bin] = 0.);
    for (k = 0; k < transformLength; k++) {

        arg = 2.*(float)bin*M_PI*(float)k / (float)transformLength;
        sinPart[bin] += inputData[k] * sign * sin(arg);
        cosPart[bin] += inputData[k] * cos(arg);

    }

}
```

Slika 2.9 Direktna realizacija diskretne Fourierove transformacije; Izvor: Stephan Bernsee's blog, 21.10.1999.

Prednost Fourierove transformacije naspram sinusne i kosinusne transformacije se i dalje treba pokazati. Prednost je ta da se ne mora baviti sinusima i kosinusima nego direktno sinusoidama. Za to je potreban dodatan korak:

```
//
// Listing 1.3: Getting sinusoid frequency, magnitude and phase from
// the Discrete Fourier Transform:
//

#define M_PI 3.14159265358979323846

long bin;
for (bin = 0; bin <= transformLength/2; bin++) {

    /* frequency */
    frequency[bin] = (float)bin * sampleRate / (float)transformLength;
    /* magnitude */
    magnitude[bin] = 20. * log10( 2. * sqrt( sinPart[bin] * sinPart[bin] +
                                             cosPart[bin] * cosPart[bin]) /
                                            (float)transformLength);

    /* phase */
    phase[bin] = 180.*atan2(sinPart[bin], cosPart[bin]) / M_PI - 90.;

}
```

Slika 2.10 Dobivanje frekvencije, amplitude i faze preko diskretne Fourierove transformacije; Izvor: Stephan Bernsee's blog, 21.10.1999.

Nakon pokretanja gore prikazanog koda na DFT izlazu, dobiva se prikaz ulaznog signala kao sume sinusoida. K-ta sinusoida je opisana uz parametar: frekvencija[k], amplituda[k], faza[k]. Važno je primijetiti da se nakon naknadne obrade (koja pretvara sinus i kosinus u sinusoidu) u kodu iznad, amplituda k-te sinusoide nazvala DFT bin amplituda, jer će se uvijek smatrati pozitivnom vrijednošću zato što se može reći da amplituda od -1 odgovara amplitude od 1 ali fazno pomaknutoj za +/- 180 stupnjeva. U literaturi, polje amplituda[] naziva se amplitudni spektar mjerenog signala, polje faza naziva se fazni spektar mjerenog signala.

Kao referencu za mjerenje bin amplitude u decibelima, od ulaznog vala se očekuje da ima uzorke vrijednosti u rasponu [-1 , 1], što odgovara amplitudi od 0dB kompletne digitalne skale(DFS-digital full scale).

7

3. PRIMJENA KRATKE FOURIEROVE TRANSFORMACIJE KOD KODIRANJA

Kao što se može vidjeti u uvodnom dijelu o osnovama Fourierove transformacije, svaki uzorkovani signal se može prikazati kombinacijom valova sinusoide koji se nazivaju segmentima vala sinusoide. Osim najosnovnijih korištenja kao što su filtriranje neželjenih frekvencija, vidjet će se da se model predstavljen sumom sinusoida može također koristiti kako bi se dobili neki drugi rezultati. Na prvi pogled čini se sasvim jasnim da ako je neki signal - koji je opisan kao suma čistih frekvencija - nije teško napraviti promjenu visine tona, no u nastavku će biti pokazano da nije baš tako.

Za razumijevanje implementacije promjene tona u frekvencijskoj domeni, u obzir se mora uzeti očigledna činjenica da se većina signal koji se susreću u praksi, kao što su oni u govoru ili u glazbi, mijenjaju ovisno o vremenu. Čak štoviše, signali koji se ne mijenjaju s vremenom zvuče dosadno i monotono i ne osiguravaju sredstva za prijenos audio informacija koje bi nešto značile. Međutim, ako se ti signali pogledaju pobliže, vidjet će se da, iako se čini da se na razne načine mijenjaju s vremenom u nekom određenom spektru, ti signali zapravo izgledaju konstantni kada se pogledaju mali uzorci tog signal, recimo nekoliko milisekundi. Dakle ove signale možemo zvati kratkotrajno stacionarnima signalima, s obzirom da su oni gotovo stacionarni unutar tog jednog uzorka od milisekunde.

Iz ovog razloga nije dobro napraviti Fourierovu transformaciju cijelog signala, s obzirom da to ne bi imalo smisla: sve promjene u spektru signala će biti gledane zajedno u prosjeku, i time mali pojedinačni dijelovi neće moći biti promatrani. Ali ako se signal podjeli na manje dijelove, analiza će u svakom trenutku vidjeti konstantan signal. Ovaj način gledanja ulaznog signala – isjeckanog na male dijelove i na svakom primijenjena diskretna Fourierova transformacija – naziva se kratka Fourierova transformacija.

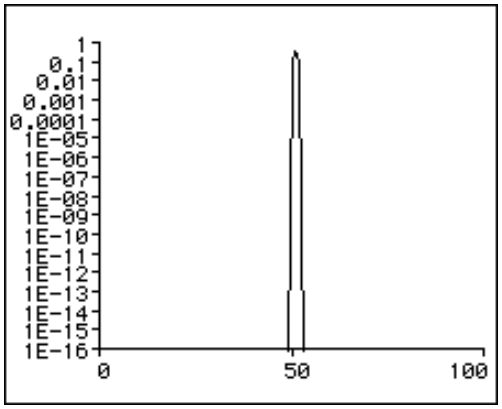
3.1. Problem razlučivosti frekvencije

Kako bi se implementirala promjena tona koristeći kratku Fourierovu transformaciju, treba se pogledati malo šire od standardne Fourierove transformacije sa njenom osnovnom sinusnom funkcijom. Pred kraj prvog dijela u kojem su objašnjene osnove Fourierove transformacije vidjelo se da se Fourierova transformacija signala procjenjuje ispitivanjem sinusoida poznatih frekvencija i mjerenjem odnosa između izmjerenog signala i reference. U dijelu rada o Fourierovoj transformaciji, izabrale su se referentne frekvencije koje će imati cjelobrojne višekratnike periode u jednom

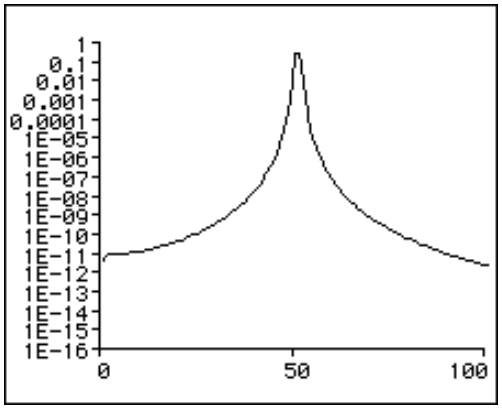
određenom dijelu diskretne Fourierove transformacije. Analogija koja se koristila je ta da se znaju referentni valovi koji se koriste kao oznake koje premošćuju prvi i zadnji uzorak u promatranom spektru. Frekvencije svih sinusoida koje se mjere biti će višekratnici inverza duljine spektra, dakle ako su naše oznake udaljene N uzoraka, kratka Fourierova transformacija će imati razmak jednak broju uzoraka podijeljenim sa brojem poznatih referentnih frekvencija. Rezultat toga je taj da se na analizu postavlja umjetna frekvencijska mreža koja zahtjeva da referentne frekvencije budu cjelobrojni višekratnici promatranog signalnog spektra određenog perioda kako bi točno stali u okvir analize.

Ovo ograničenje neće imati posljedice za frekvenciju signala koji se ovdje promatra, koji je smješten točno na te referentne frekvencije (uvijek će se savršeno preklapati), ali s obzirom da se treba riješiti problem kod signala u svakodnevici ne može se očekivati da će signali uvijek ispuniti ovaj uvjet. Čak štoviše, vjerojatnost da će se mjereni signal poklapati sa referentnim signalom je jako malena

Mora se dakle razmotriti što se događa sa frekvencijama koje se nalaze između očekivanih vrijednosti. Tu se treba spomenuti efekt "rastezanja" signala, tj. signali koji imaju neku vrijednost između onih očekivanih će najveći doprinos dati onima kojima su – frekvencijski gledano – najbliže, ali će isto tako dio doprinosa ići i susjednom. Slika 3.2 koja je prikazana ispod prikazuje kako će izgledati amplitudni spektar u ovom slučaju.



Slika 3.1 Oblik mjerenog signala čija će se frekvencija preklapati sa bin frekvencijom ; Izvor: Stephan Bernsee's blog, 21.10.1999.



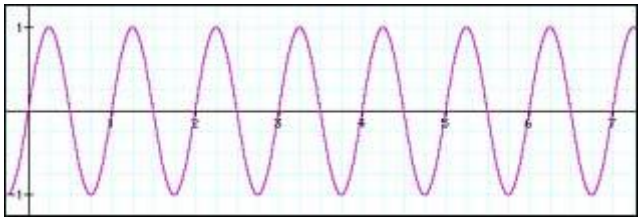
Slika 3.2 Oblik mjerenog signala čija se frekvencija neće preklapati sa bin frekvencijom; Izvor: Stephan Bernsee's blog, 21.10.1999.

Kao što se može vidjeti iz Slike 3.1, kada se mjereni signal poklapa sa bin frekvencijom, doprinijeti će samo tom bin-u. Ako nije točno centriran na jednu od bin frekvencija, njegova veličina će biti raširena na susjedne bin-ove, a to je razlog zašto na Slici 3.2 ima jako široku bazu dok na Slici 3.1 ima samo vrh na bin-u 50.

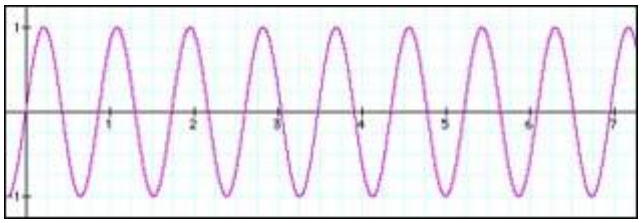
Razlog iz kojega se ovo naglašava je taj što je ovaj efekt koji se događa jedna od glavnih prepreka većini ljudi koji pokušavaju implementirati promjenu frekvencije koristeći kratku Fourierovu transformaciju. Glavni problem s ovim efektom nije čak ni veličina spektra, s obzirom da nam veličina spektra govori samo to da je određena frekvencija prisutna u tom signal koji je promatran. Glavni problem je, a to će se dalje kroz rad vidjeti, je faza bin-a.

3.2. Poveznica faze i frekvencije

U prvom dijelu rada je spomenuto da je sinusni signal uz pravilnu naknadnu obradu opisan frekvencijom, fazom, amplitudom. Ove tri veličine opisuju sinusoidu u bilo kojem trenutku za dani transformacijski okvir. Također je bilo prikazano da je frekvencija određena mrežom na kojoj ispitujemo i uspoređujemo signal sa referentnim signalom. Dakle, bilo koja dva bina će uvijek biti udaljena za *sampleRate/N* (broj uzoraka / broj poznatih ref. frekvencija) hertza (gledano sa strane frekvencije). Gore se moglo vidjeti da u slučajevima u kojima se mjereni signal poklapa sa bin frekvencijom sve ide glatko – očito je da će imati frekvenciju koja je višekratnik *sampleRate/N*. Međutim, treba se zapitati što se može očekivati kada ta frekvencija nije višekratnik *sampleRate/N* (frekvencijski gledano).

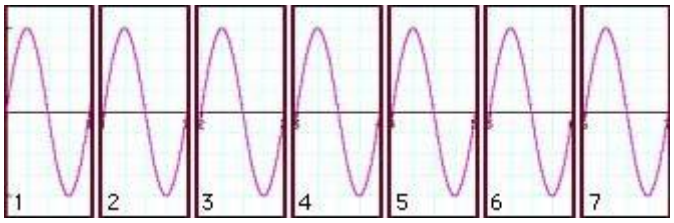


Slika 3.3 Valni oblik mjerenog signala koji ima frekvenciju jednaku onoj koju ima bin; Izvor: Stephan Bernsee's blog, 21.10.1999.

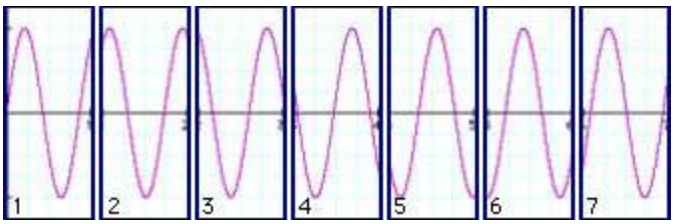


Slika 3.4 Valni oblik mjerenog signala koji nema frekvenciju jednaku onoj koju ima bin; Izvor: Stephan Bernsee's blog, 21.10.1999.

Ova dva grafa izgledaju poprilično normalno, osim što se vidi da ova dva signala nemaju istu frekvenciju – valni oblik na drugoj slici ima višu frekvenciju od valnog oblika prikazanog na prvoj. Već je prije naglašeno da će se koristiti mali okviri za analizu signal. Kada se ta dva signala podijeli u te okvire to izgleda ovako:



Slika 3.5 Signal koji je bio prikazan na slici 3.3 sada je podijeljen na 7 dijelova; Izvor: Stephan Bernsee's blog, 21.10.1999.



Slika 3.6 Signal koji je bio prikazan na slici 3.4 sada je podijeljen na 7 dijelova; Izvor: Stephan Bernsee's blog, 21.10.1999.

Svaki dio pojedinačno će biti analiziran i proslijeđen transformaciji. Vidi se da Signal na Slici 3.5 normalno odgovara svakom okviru, dok se kod Slike 3.6 vidi da postoji fazni pomak u svakom okviru. Uzorkovani signal koji se vidi u Slici 3.3 koji je točno centriran na bin frekvenciji, jasno se dijeli na sedam uzastopnih okvira analize. U svakom okviru valni oblik počinje na nuli i završava na nuli. Točnije rečeno: u svakom okviru mjereni signal započinje svoj ciklus na istom mjestu, tj. počinje sa istom fazom.

Uzorkovani signal koji se vidi u Slici 3.4, koji se nalazi negdje između dvije bin frekvencije, ne može se lijepo podijeliti na 7 uzastopnih analitičkih okvira. U svakom okviru valni oblik ima jasno vidljiv fazni pomak tj. započinje svoj ciklus u drugoj točki. Što je mjerena frekvencija udaljenija od bin frekvencije, to će fazni pomak biti veći. Dakle, ono što se iz ovoga može zaključiti je to da valni oblici čija je frekvencija točno na bin frekvenciji nemaju fazni pomak, ili uvijek imaju isti fazni pomak, tj. uvijek započinju svoj ciklus na istom mjestu, a valni oblici koji se nalaze negdje između dvije bin frekvencije će u svakom analitičkom okviru imati drugačiji fazni pomak. Novi zaključak koji se izvodi je taj da razlika u fazama između dva analitička okvira ukazuje na to da postoji frekvencijsko odstupanje od bin frekvencije.

Drugim riječima: ako se promatrana sinusoida (jedan segment cijelog signala) mjeri sa njezinim pripadajućim bin veličinama: amplitudom, frekvencijom i fazom, amplituda će označiti do koje granice je točno ta frekvencija zastupljena u ukupnom signalu kojeg se promatra. Frekvencija će poprimiti svoju bin frekvenciju, a faza će se mijenjati sukladno svom odstupanju od te bin frekvencije. S obzirom da je do sad već utvrđeno da promjena faze u svakom analitičkom okviru znači odstupanje frekvencije od bin frekvencije, može se isto tako koristiti fazni pomak kako bi se izračunala prava frekvencija sinusoide. Ovo sve vodi zaključku da se tri veličine koje se dobiju analizom svake sinusoide – bin amplituda, bin frekvencija, bin faza – mogu svesti na amplitudu i pravu frekvenciju.

Matematički gledano, računanje promjene parametara naziva se oduzimanje (u slučaju funkcije to računanje se naziva derivacija funkcije). To je ono što je u ovom slučaju potrebno zato što se treba izračunati razlika između trenutačne vrijednosti parametra i posljednje vrijednosti parametra, tj. koliko se parametar promijenio od zadnjeg mjerenja. U ovdje promatranom slučaju to je bin faza. Dakle može se reći da je odstupanje k-tog dijela od bin frekvencije proporcionalno odstupanju faze. Kasnije u radu ovo će se saznanje koristiti kako bi se izračunala prava frekvencija sinusoide koja je segment cijelog signala.

3.3. Faktor preklapanja i njegov utjecaj na signal

Postoji još jedna stvar na koju se mora obratiti pozornost. Jednostavno dijeljenje signala na uzastopne, ne-preklapajuće okvire kao što je to opisano u gornjem primjeru nije dovoljno. Mora se koristiti preklapajuće okvire iz nekoliko razloga među kojima je prvi zrcaljenje koje se radi kako bi se smanjilo rastezanje bin amplitude, te kako bi se moglo napraviti jasnu razliku, tj. kako bi se moglo jasno izdvojiti derivaciju bin faze. Standardni faktor preklapanja iznosi 4, tj. dva susjedna prozora se preklapaju barem 75%. Na svu sreću značajnost preklapanja okvira kod bin faze i njenih derivacija je lako za izračunati i može biti izuzeto kod računanja prave bin frekvencije. Primjer ovoga bit će prikazan u kodu danom ispod. To je zapravo vrlo jednostavno za izračunati, s obzirom da se samo treba izračunati koliko daleko će promatrana derivacija bin faze napredovati - s obzirom na dano preklapanje - i onda oduzeti to odstupanje od fazne razlike, a to sve prije računanja prave frekvencije k-tog dijela.

Druga, važnija stvar povezana s ovime, koja se mora uzeti u obzir je ta da izbor preklapanja utječe na način na koji se izdvajaju prave frekvencije pojedinih segmenata. Ako se promatra okvire koji se velikim dijelom preklapaju, područje u kojem prava frekvencija svake sinusoide može varirati biti će veliko. Ako se izabere manje preklapanje, to područje variranja biti će manje.

Kako bi se uvidjelo zašto je to tako, prvo se treba utvrditi kako se zapravo mjeri bin frekvencija. Kao što se može vidjeti iz danog koda koristi se *arctg* omjer sinusa i kosinusa (u primjeru koda oni se nazivaju imaginarnim (Im) i realnim (Re) dijelom svake bin frekvencije zbog matematičke terminologije). Predznak sinusa i kosinusa određuje kvadrant u kojem se fazni kut mjeri. Koristeći to saznanje može se pretpostaviti da će bin faza u svakom trenutku biti između pozitivne ili negativne vrijednosti, s obzirom da je to opseg vrijednosti koju vraća funkcija *atan2()*.

Dakle, povećanje faze između bilo koja dva okvira ima maksimum. Maksimum tog povećanja postoji za negativnu razliku kutova - koja označava negativno odstupanje od bin frekvencije - te za pozitivnu razliku kutova - koja označava pozitivno odstupanje od bin frekvencije. Kako bi se osiguralo da je promatrana frekvencija centrirana oko nule (tj. da bude mjerena kao apsolutna vrijednost s obzirom na ishodište (npr. u intervalu npr. -50000 do +50000) koje je potrebno kako bi mjerili frekvencijsko odstupanje) vrijednost promatrane fazne razlike vraća se u +/- interval. Ovo je potrebno s obzirom da fazno odstupanje (koje se izuzima zbog preklapanja) može uzrokovati da fazna razlika bude izvan toga intervala.

U nekom jednostavnom slučaju kada se dva susjedna okvira ne bi preklapala i kada se zna +/- interval kako bi se označilo odstupanje frekvencije jednog dijela od promatrane bin frekvencije, moglo bi se izdvojiti frekvencijsko odstupanje jedino od +/- 0.5 binova s obzirom da je to maksimalni napredak u fazi između dva okvira koji se mogu jasno izdvojiti u ovom slučaju. Kada frekvencija mjerenog signala prijeđe granicu između dva susjedna bina kratke Fourierove transformacije, razlika između faza će se vratiti na početak promatranog intervala, a frekvencija tih pojedinih dijelova biti će daleko od stvarne frekvencije ulaznog sinusnog signala.

Tablica 3.1. Odstupanje frekvencija od bin frekvencija za bin=112.0; Izvor: Stephan Bernsee's blog, 21.10.1999.

2411.718750 Hz (bin 112.0):

Broj Bin-a	Bin frekvencija [Hz]	Bin amplituda	Bin fazna razlika	Približna prava frekvencija [Hz]
110	2368.652344	0.000000	-0.403069	2367.270980
111	2390.185547	0.500000	0.000000	2390.185547
112	2411.718750	1.000000	0.000000	2411.718750
113	2433.251953	0.500000	0.000000	2433.251953
114	2454.785156	0.000000	0.112989	2455.172383

Tablica 3.2. Odstupanje frekvencija od bin frekvencija za bin=112.2; Izvor: Stephan Bernsee's blog, 21.10.1999.

2416.025391 Hz (bin 112.2):

Broj Bin-a	Bin frekvencija [Hz]	Bin amplituda	Bin fazna razlika	Približna prava frekvencija[Hz]
110	2368.652344	0.022147	1.256637	2372.958983
111	2390.185547	0.354352	1.256637	2394.492187
112	2411.718750	0.974468	1.256637	2416.025391
113	2433.251953	0.649645	1.256637	2437.558594
114	2454.785156	0.046403	1.256637	2459.091797

Tablica 3.3. Odstupanje frekvencija za od bin frekvencija za bin=112.49; Izvor: Stephan Bernsee's blog, 21.10.1999.

2422.270020 Hz (bin 112.49):

Broj Bin-a	Bin frekvencija [Hz]	Bin amplituda	Bin fazna razlika	Približna prava frekvencija [Hz]
110	2368.652344	0.024571	3.078761	2379.203614
111	2390.185547	0.175006	3.078761	2400.736816
112	2411.718750	0.854443	3.078761	2422.270020
113	2433.251953	0.843126	3.078761	2443.803223
114	2454.785156	0.164594	3.078761	2465.336426

Vidi se da kad se krene točno iz frekvencije bin 112 kao što je pokazano u Tablici 3.1, prave frekvencije svih značajnih kanala (amplituda $\neq 0$) su točno centrirane na njihove bin frekvencije, što se i moglo očekivati s obzirom da se koristila frekvencija koja točno odgovara. Vrijednost od 0.5 za amplitude bin-a 111 i 113 je zbog zrcaljenja koje se koristi. Frekvencija mjerenog signala iz Tablice 3.2 može se povećati da bude 20% udaljenija od 112-te bin frekvencije. Vidi se da bin 112 točno prati signal, dok bin iznad njega, prema kojem se frekvencija kreće, ide još dalje od točne frekvencije iako njegova amplituda raste. To je zato što se nalazi u svome intervalu i zapravo ide u suprotnome smjeru. To se još jasnije može vidjeti u Tablici 3.3 koja pokazuje da bin 113 (koji bi - s obzirom na svoju amplitudu - zapravo trebao biti bliže pravoj frekvenciji mjerenog signala) ide još više u neželjenom smjeru.

Tablica 3.4. Odstupanje frekvencija za od bin frekvencija za bin=112.5; Izvor: Stephan Bernsee's blog, 21.10.1999.

2422.485352 Hz (bin 112.5):

Broj Bin-a	Bin frekvencija [Hz]	Bin amplituda	Bin fazna razlika	Približna prava frekvencija [Hz]
110	2368.652344	0.024252	-3.141593	2357.885742
111	2390.185547	0.169765	-3.141593	2379.418945
112	2411.718750	0.848826	-3.141593	2400.952148
113	2433.251953	0.848826	-3.141593	2422.485352
114	2454.785156	0.169765	-3.141593	2444.018555

U Tablici 3.4 frekvencija mjerenog signala je na pola puta između dva bina i to između 112 i 113. To se može vidjeti iz bin amplitude koja ukazuje na ovu činjenicu time što ima istu vrijednost za oba bina. Sada bin 113 zauzima pravu frekvenciju dok se bin 112 vraća na početak svog intervala (od 3.079 ≈3.1415 [+?] do 3.1415 [-?]). Usporedbe radi, evo kako izgledaju signali kada se koristi preklapanje 75%

Tablica 3.5. Odstupanje frekvencija za od bin frekvencija za bin=112.5, preklapanje 4x; Izvor: Stephan Bernsee's blog, 21.10.1999.

2422.485352 Hz (bin 112.5), 4x overlap:

Broj Bin-a	Bin frekvencija [Hz]	Bin amplituda	Bin fazna razlika	Približna prava frekvencija [Hz]
110	2368.652344	0.024252	-2.356196	2336.352516
111	2390.185547	0.169765	2.356194	2422.485348
112	2411.718750	0.848826	0.785398	2422.485352
113	2433.251953	0.848826	-0.785398	2422.485351
114	2454.785156	0.169765	-2.356194	2422.485355
115	2476.318359	0.024252	2.356196	2508.618186

Kada se želi izmijeniti signal (to je ono što se treba napraviti kako bi se postigao efekt promjene frekvencije), može se vidjeti da će se bez preklapanja naići na slučaj (prikazan u Tablici 3.4) u kojem se kod ponovne sinteze pojavljuju dvije sinusoide jednake amplitude, ali koje su međusobno udaljene za frekvenciju jednog bin-a. U ovom slučaju ta bi razlika iznosila oko 21.5Hz. Dakle, kada bi se promatrani signal ponovno sastavljao dobilo bi se dvije sinusoide međusobno razmaknute za 21.5Hz, dok je na ulazu bila samo jedna sinusoida. Upravo zato nije iznenađenje što ovaj sintezirani, tj. spojeni signal neće zvučati kao originalni signal. Naravno ako bi se ponovno sastavljao signal bez promjene frekvencije, može se očekivati da će se ovi efekti poništiti. Međutim, to neće biti tako kada se izmijeni promatrani signal skaliranjem dijelova frekvencije – moglo bi se očekivati da će ovo dovesti do jasne greške u promatranom signalu s obzirom da će se uklapanje sada dogoditi u drugoj stopi. Kada se pogleda Tablica 3.5 u kojoj se koristi 4x preklapanje (tj. 75%), vidi se da sve susjedne sinusoide sve do onih koje imaju bin amplitudu 0.17 (oko -15dB) imaju približno istu frekvenciju, te se greška može praktički zanemariti. Sljedeća sinusoida koja nema točnu frekvenciju je oko -32dB niže, što je puno bolje nego u slučaju kad nije bilo preklapanja. Dakle, očekuje se da će ovo zvučati mnogo bolje.

Iz do sada rečenih stvari vidi se da se izabiranjem većeg preklapanja, što je zapravo pretjerano uzorkovanje kratke Fourierove transformacije u vremenu, povećava mogućnost da se približno

odredi pravu frekvencija mjerenog sinusnog signal tako što će se povećati raspon u kojem svaka sinusoida može odstupati od svoje bin frekvencije. Raspon od +/- 0.5 periode ima drugačije značenje kada se okvire smjesti bliže, kao što fazna razlika za 2x i 4x preklapanja znači dvostruko veće frekvencijsko odstupanje za slučaj 4x preklapanja u usporedbi s onim koje je tu kada imamo 2x preklapanje. Dakle što se manje razmakne okvire, tj. što se više oni preklapaju, to će se lakše odrediti prave frekvencije sinusoide ulaznog, mjerenog signal. To će se odraziti na računanje na taj način da će se morati izvesti duplo više kratkih Fourierovih transformacija kada se preklapanje povisi sa 50% na 75%. Što se tiče posljedica s obzirom na praktičnu primjenu, sada se jasno vidi zašto je potrebno preklapanje koje je dovoljno veliko: ako se mjeri signal čija je frekvencija između dva bina, bit će dva bina koja će imati veliku amplitudu. Međutim, s obzirom da je prava frekvencija negdje između dva bina, a svaki bin može odstupati točno određeni iznos (zavisno o preklapanju), postoji mogućnost da će biti dvije istaknute sinusoide koje će imati različite, ali približno slične frekvencije. Još gore, ako je prava frekvencija između dva bin-a recimo k i k+1 kao što je prikazano u primjeru u Tablici 3.4, frekvencija sinusoide k će se udaljiti od prave frekvencije zato što će njena fazna razlika – pokušavajući se držati frekvencije ulaznog signal – priviti uz suprotnu granu. To će proizvesti zvučni udar u ponovno sinteziranom (sastavljenom) zvuku i time oštetiti rezultat. 75% preklapanje rješava sljedeće pitanje: amplitude susjednih sinusoida su smanjene na prihvatljivu vrijednost i mogu se priviti uz pravu frekvenciju bez uklapanja – postale se frekvencijski fleksibilnije što daje puno bolju zvučnu kvalitetu, a rezultat svega toga je da je zvučni udar gotovo nestao.

3.4. Promjena frekvencije

Nakon što su savladane poteškoće vezane za računanje prave frekvencije pojedinih dijelova za binove, promjena frekvencije je lakša. Sada treba pogledati što se dobilo nakon što se izračunala prava frekvencija dijelova.

Postoji određeni niz brojeva koji sadrže vrijednost iznosa amplitude. Kada se frekvencija skalira da bude “oštra”, očekuje se da se amplitudni niz proširi - prema faktoru promjene frekvencije – prema kraju više frekvencije. Isto tako kada skaliramo frekvenciju da postane što ravnija očekuje se da spektar amplituda teži prema kraju niže frekvencije. Stvarne vrijednosti amplitude pohranjene u nizu ne bi se trebale mijenjati, kao što se čisti sinusni signal od -2dB i 1000Hz očekuje da pri promjeni frekvencije od 0.5x ostane -2dB ali uz frekvenciju 500Hz.

Također postoji niz u koji su pohranjene prave frekvencije pojedinih segmenata signala. Kao što je to bilo i s amplitudnim spektrom i od frekvencijskog spektra se očekuje da se prilagođava prema

određenom faktoru. Međutim, za razliku od amplitudnih vrijednosti, vrijednosti koje su pohranjene u frekvencijski niz označavaju pravu frekvenciju naših malih dijelova sinusoide. Zbog toga se očekuje da se i oni također mijenjaju prema faktoru promjene frekvencije.

U najboljem slučaju, može se staviti skalirane frekvencije segmenata signala na mjesto gdje pripadaju zajedno sa pripadajućim amplitudama. Ovo će sačuvati većinu energije sadržane u ulaznog signalu.

Također se vidi zašto promjena frekvencije ovom metodom automatski uključuje anti-aliasing: jednostavno se ne računaju binovi iznad Nyquistove frekvencije tako da se zaustavi *FFTFrameSize2*, a za to nije potreban dodatni korak.

3.5. Amplitudni i frekvencijski spektar

Kako bi se dobio očekivani izlazni signal, treba se vratiti korak po korak nazad kako bi se dobio amplitudni i frekvencijski spektar. Za pretvaranje amplitudnog i frekvencijskog prikaza u bin amplitude, bin frekvenciju i bin fazu, prati se isti put kojim se došlo do ovog koraka. S obzirom da su sinusni i kosinusni dijelovi kratke Fourierove transformacije periodični i definirani za svaki trenutak, ne samo za neki određeni interval, ne treba se posebno brinuti za fazu u ovom procesu vraćanja – to se događa automatski. Nakon obavljanja inverzne Fourierove transformacije, treba se nanizati preklapajuće okvire istim odabirom koraka kako bi se dobio nazad signal koji je bio frekvencijski pomaknut.

3.6. Zapisivanje u kod

Sada se postavlja pitanje kako sve ovo implementirati program. Prvo se treba dobiti trenutni okvir brze Fourierove transformacije iz *FIFO* reda.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace PitchShifter
{
    public class PitchShifter
    {

        #region Private Static Memebers
        private static int MAX_FRAME_LENGTH = 16000;
        private static float[] gInFIFO = new float[MAX_FRAME_LENGTH];
        private static float[] gOutFIFO = new float[MAX_FRAME_LENGTH];
        private static float[] gFFTworksp = new float[2 * MAX_FRAME_LENGTH];
        private static float[] gLastPhase = new float[MAX_FRAME_LENGTH / 2 + 1];
        private static float[] gSumPhase = new float[MAX_FRAME_LENGTH / 2 + 1];
        private static float[] gOutputAccum = new float[2 * MAX_FRAME_LENGTH];
        private static float[] gAnaFreq = new float[MAX_FRAME_LENGTH];
        private static float[] gAnaMagn = new float[MAX_FRAME_LENGTH];
        private static float[] gSynFreq = new float[MAX_FRAME_LENGTH];
        private static float[] gSynMagn = new float[MAX_FRAME_LENGTH];
        private static long gRover, gInit;
        #endregion
```

Slika 3.7. FIFO okvir.

To se traži zato što se želi izbjeći točno određivanje veličine ulaznog buffera. Na ovaj način mogu se procesuirati svi podatci nezavisno o veličini okvira brze Fourierove transformacije koji se koristi za promjenu visine tona.

Podaci su zatim zrcaljeni:

```
/* do windowing and re,im interleave */
for (k = 0; k < fftFrameSize; k++)
{
    window = -.5 * Math.Cos(2.0 * Math.PI * (double)k / (double)fftFrameSize) + .5;
    gFFTworksp[2 * k] = (float)(gInFIFO[k] * window);
    gFFTworksp[2 * k + 1] = 0.0F;
}
```

Slika 3.8. Zrcaljenje.

i naizmjenično pohranjeni u *gFFTworksp* polje,

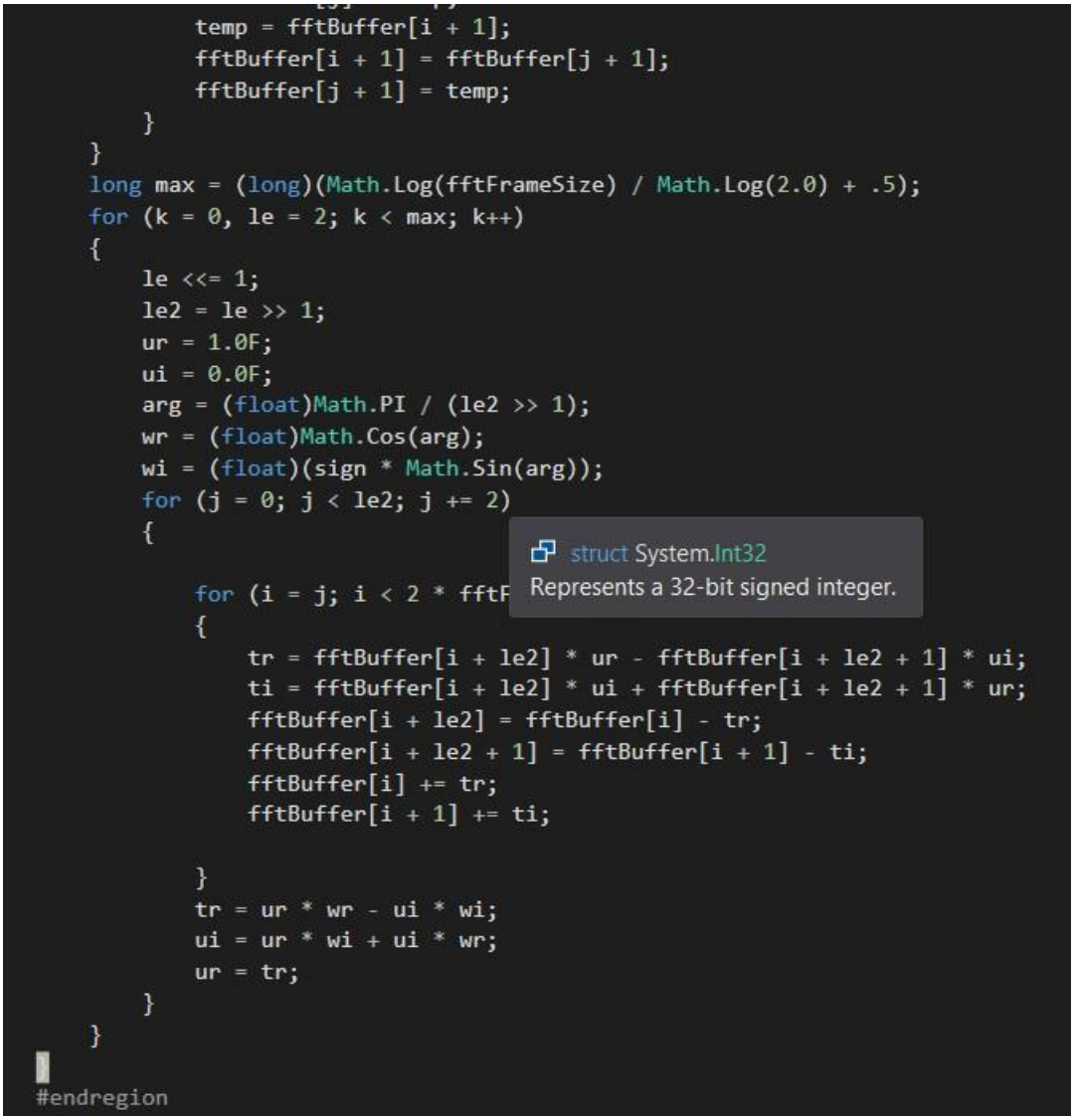
```
using System;
using System.Collections.Generic;
using System.Text;

namespace PitchShifter
{
    public class PitchShifter
    {

        #region Private Static Memebers
        private static int MAX_FRAME_LENGTH = 16000;
        private static float[] gInFIFO = new float[MAX_FRAME_LENGTH];
        private static float[] gOutFIFO = new float[MAX_FRAME_LENGTH];
        private static float[] gFFTworksp = new float[2 * MAX_FRAME_LENGTH];
        private static float[] gLastPhase = new float[MAX_FRAME_LENGTH / 2 + 1];
        private static float[] gSumPhase = new float[MAX_FRAME_LENGTH / 2 + 1];
        private static float[] gOutputAccum = new float[2 * MAX_FRAME_LENGTH];
        private static float[] gAnaFreq = new float[MAX_FRAME_LENGTH];
        private static float[] gAnaMagn = new float[MAX_FRAME_LENGTH];
        private static float[] gSynFreq = new float[MAX_FRAME_LENGTH];
        private static float[] gSynMagn = new float[MAX_FRAME_LENGTH];
        private static long gRover, gInit;
        #endregion
```

Slika 3.9. gFFTworksp polje.

gdje se uz pomoć algoritma brze Fourierove transformacije *ShortTimeFourierTransform()* transformiraju.

```
public static void ShortTimeFourierTransform(float[] fftBuffer, long fftFrameSize, long sign)
{
    float wr, wi, arg, temp;
    float tr, ti, ur, ui;
    long i, bitm, j, le, le2, k;

    for (i = 2; i < 2 * fftFrameSize - 2; i += 2)
    {
        for (bitm = 2, j = 0; bitm < 2 * fftFrameSize; bitm <<= 1)
        {
            if ((i & bitm) != 0) j++;
            j <<= 1;
        }
        if (i < j)
        {
            temp = fftBuffer[i];
            fftBuffer[i] = fftBuffer[j];
            fftBuffer[j] = temp;
            temp = fftBuffer[i + 1];
            fftBuffer[i + 1] = fftBuffer[j + 1];
            fftBuffer[j + 1] = temp;
        }
    }
    long max = (long)(Math.Log(fftFrameSize) / Math.Log(2.0) + .5);
    for (k = 0, le = 2; k < max; k++)
    {
        le <<= 1;
```

Slika 3.10. Fourier algoritam 1. dio.



Slika 3.11. Fourier algoritam 2. dio.

Sada je tu sve što je potrebno za pozitivne i negativne frekvencije binova digitalne Fourierove transformacije. Koristiti će se samo pozitivne jer izvorno korišteni signal ima samo realnu komponentu. Onda se te pozitivne frekvencije pretvaraju u amplitude i faze koristeći pravokutno-polarnu transformaciju i time se dobiva trenutna bin frekvencija zbog fazne razlike između dva susjedna okvira kratke Fourierove transformacije. Iz toga se može dobiti fazni pomak između dva susjedna okvira koji se kompenzira kako bi se dobio očekivani, o frekvenciji ovisan fazni pomak. Također se dobije i prava parcijalna frekvencija. Nakon što se dovrši proces promjene visine tona, treba se vratit iz frekvencijskog područja u novonastali slijed vremenskog područja. Nakon isprepletanja [Re, Im] polja, zrcaljenja, i ponovnog skaliranja, podatci se stavljaju u izlazni red kako bi se osiguralo da postoji broj izlaznih podataka jednak broju ulaznih podataka. *inFifoLatency* su

uzorci koji predstavljaju globalno ulazno/izlazno kašnjenje (broj sample-ova tišine početnog izlaza). To mora biti uzeto u obzir kada se podatci zapisuju u datoteku.

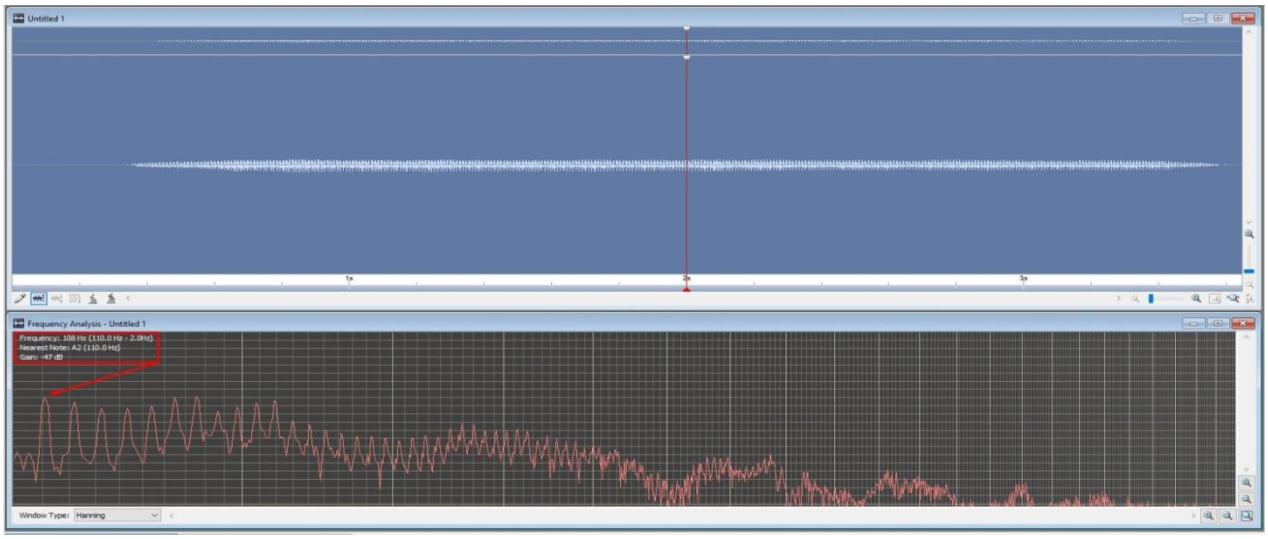
```
long i, k, qpd, index, inFifoLatency, stepSize, fftFrameSize2;


float[] outdata = indata;
/* set up some handy variables */
fftFrameSize2 = fftFrameSize / 2;
stepSize = fftFrameSize / osamp;
freqPerBin = sampleRate / (double)fftFrameSize;
expct = 2.0 * Math.PI * (double)stepSize / (double)fftFrameSize;
inFifoLatency = fftFrameSize - stepSize;
if (gRover == 0) gRover = inFifoLatency;


/* main processing loop */
for (i = offset; i < sampleCount; i++)
{

    /* As long as we have not yet collected enough data just read in */
    gInFIFO[gRover] = indata[i];
    outdata[i] = gOutFIFO[gRover - inFifoLatency];
    gRover++;

    /* now we have enough data for processing */
    if (gRover >= fftFrameSize)
    {
        gRover = inFifoLatency;
```

Slika 3.12. Kašnjenje.

U parametrima rutine treba primjetiti: *routinesmbPitchShift()* uzima *pitchshiftfactor* koji se nalazi između 0.5 (jedna oktava niže) i 2 (jedna oktava višlje). Ako je vrijednost 1, visina tona se neće promijeniti. Ako je potrebno imati širi raspon promjene visine tona, treba se malo podesiti kod. *numSampsToProcess* pokazuje rutini kolikom broju uzoraka iz *indata[0 ... numSampsToProcess-1]* bi se trebala promijeniti visina i koliki broj treba biti premješten u *outdata[0 ... numSampsToProcess-1]*. To može biti bilo koji broj uzoraka. Dva buffera mogu biti jednaka (jer se podatci procesuiraju trenutno). *fftFrameSize* definira veličinu okvira brze Fourierove transformacije koji se koristi za obradu. Standardne vrijednosti su 1024, 2048, 4096. To zapravo može biti bilo koja vrijednost manja ili jednaka *MAX_FRAME_LENGTH*, ali mora biti na drugu potenciju osim ako se koristi brza Fourierova transformacija koja može rješavati i ostale veličine okvira.

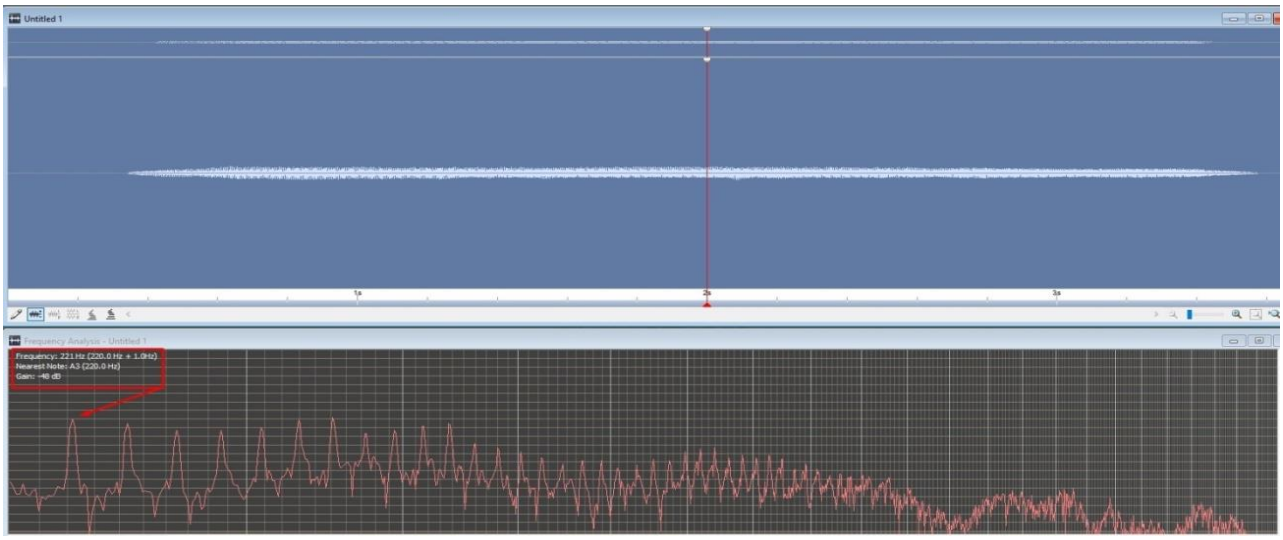
3.7. Rezultat transformacije

Kako bi se rezultati što bolje prikazali korišten je program koji se služi za analizu zvukova, jer u sebi ima alate koji omogućavaju viualizaciju svih frekvencija koje se u određenom trenutku pojavljuju. Program se zove WavePad. Kako bi se točno moglo pratiti što se događa, snimljen je ton za koji će se znati kolika mu je frekvencija i koliku bi frekvenciju trebao imati kada se – u ovom slučaju je tako napravljeno – povisi za jednu oktavu. Frekvencija je u oba slučaja promatrana u istom trenutku. Na Slici 3.13. prikazana je frekvencija tona A2:



Slika 3.13. Frekvencija tona A2.

Frekvencija tona A2 je 110Hz, a nakon što je napravljena transformacija, dobivena je frekvencija tona A3, tj. frekvencija od 220Hz



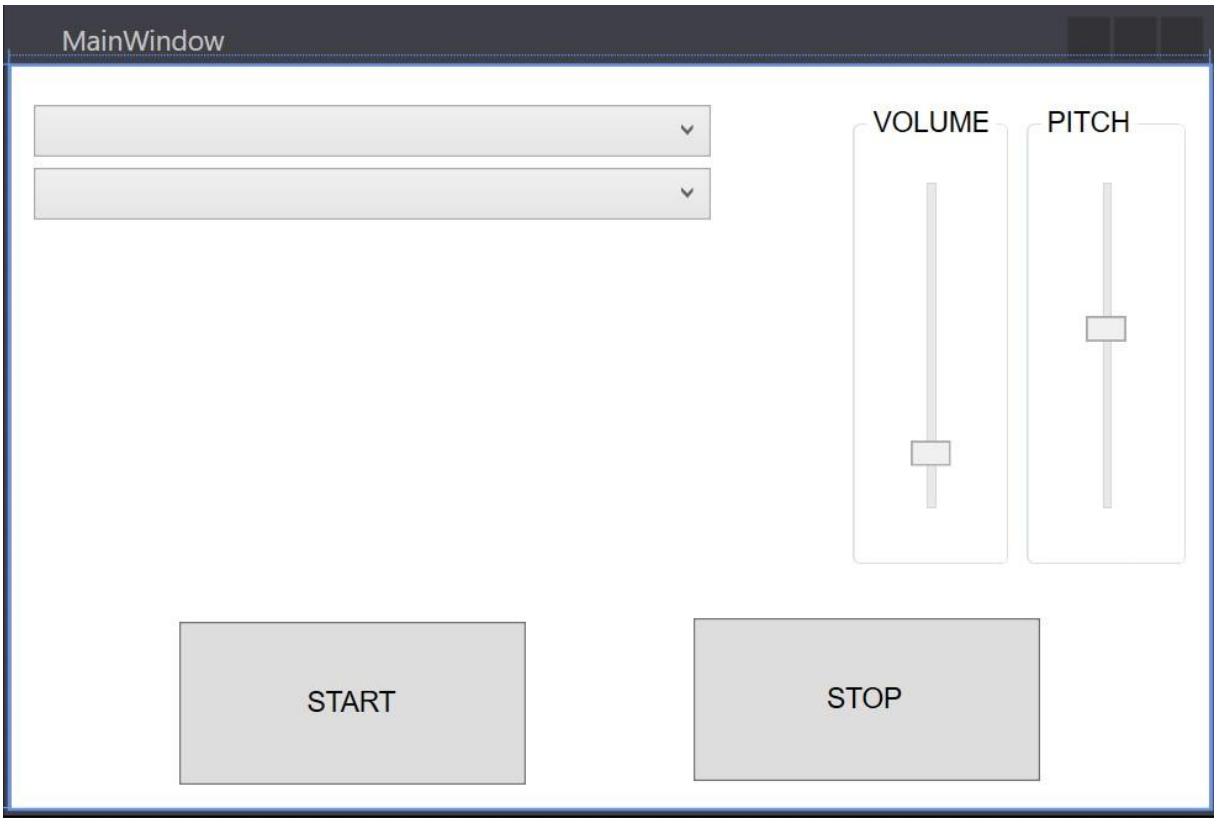
Slika 3.14. Frekvencija tona A3.

4. C# I OSTALI KORIŠTENI ALATI

C# je objektno orijentirani programski jezik kojega su razvili Anders Hejlsberg i drugi u tvrtki Microsoft. C# je izumljen s ciljem da .NET platforma dobije programski jezik, koji bi maksimalno iskoristio njezine sposobnosti. Sličan je programskim jezicima Java i C++.

Aplikacija koristi CSCore biblioteku koja omogućava spremanje, reproduciranje i procesuiranje audio signala. Također je korišten algoritam napisan od strane Stephan Bernsee-a, te klase „*SampleDSP*“, „*SimpleMixer*“ i „*PitchShifter*“.

Vizualizacija je napravljena u WPF-u (Windows PresentationFoundation). Velik dio programera koristi WF (Windows Forms), ali ga većina koristi jer je stariji i ima puno literature za njega. U ovom radu korišten je WPF zato jer omogućava veću kontrolu nad svim komponentama vizualizacije. Sučelje je prikazano na Slici 4.1.

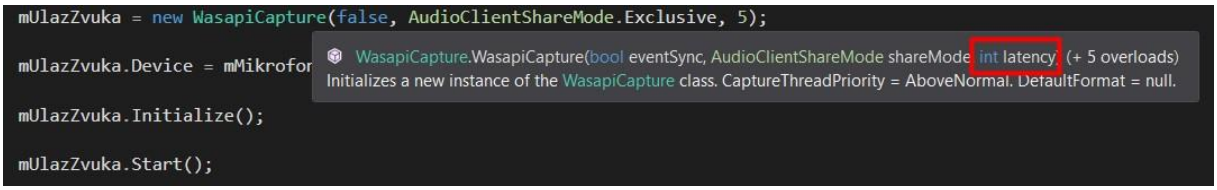


Slika 4.1. Izgled korisničkog sučelja.

Slika 4.2 prikazuje XAML kod kojim je dobivena vizualizacija prikazana na Slici 4.1

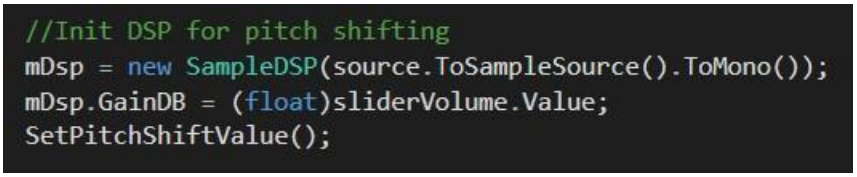
Slika 4.2. XAML kod korisničkog sučelja.

Kada se klikne na start, aplikacija inicijalizira mikrofon u *exclusive mode* kako bi se dobilo što manje kašnjenje. To omogućava da zvuk koji se vraća kroz zvučnike minimalno kasni.



Slika 4.3. Smanjivanje kašnjenja.

Zatim aplikacija stvara SampleDSP objekt te postavlja vrijednosti glasnoće i visine tona.



Slika 4.4. Postavka glasnoće i visine tona

Zatim treba stvoriti mixer u koji će se dodati DSP objekt

```
//Init mixer
mMixer = new SimpleMixer(1, 44100) //mono, 44,1 KHz
{
    FillWithZeros = false,
    DivideResult = true //This is set to true for avoiding tick sounds because of exceeding -1 and 1
};

//Add our sound source to the mixer
mMixer.AddSource(mDsp.ChangeSampleRate(mMixer.WaveFormat.SampleRate));
```

Slika 4.5. Mixer kod.

Posljednji korak je inicijalizacija audio uređaja koji emitira signal primljen na mikrofonu.

```
// Inicijalizacija audio uređaja uz kašnjenje od 5ms
mIzlazZvuka = new WasapiOut(false, AudioClientShareMode.Exclusive, 5);

mIzlazZvuka.Device = mZvucnici[comboBoxZvucnici.SelectedIndex];

mIzlazZvuka.Initialize(mMixer.ToWaveSource(16));

// Start
mIzlazZvuka.Play();
return true;
```

Slika 4.6. Inicijalizacija audio uređaja.

5. ZAKLJUČAK

U ovom radu pojašnjene su osnove transformacije zvuka gdje je objašnjeno kako se može dobiti određeni željeni oblik signala koristeći određeni broj sinusoida uz određeni recept. Broj sinusoida određen je brojem uzoraka koji su odabrani, a za pisanje točnog recepta potrebno je poznavati također i amplitudu pojedinog segmenta sinusoide koja se dobije tako da se uzme referentni sinusni oblik poznate frekvencije i jedinične amplitude i to se onda pomnoži sa uzorcima signala. Za upravljanje zvukom se koristilo kratku Fourierovu (ulazni signal isjeckan na male dijelove i na svakom primijenjena diskretna Fourierova transformacija) i inverznu Fourierovu transformaciju kako bi se ugradila promjena visine tona i uključilo prikladno anti-preklapanje u frekvencijskom području. Objašnjeno je koji problemi nastaju kada se promatrana frekvencija nalazi između referentnih frekvencija i tu je spomenut efekt rastezanja signala (signali koji imaju neku vrijednost koja se nalazi između onih koje su očekivane će najveći doprinos dati onima kojima su – frekvencijski gledano – najbliže, ali će isto tako dio doprinosa ići i susjednom). Prikazano je kako izgleda promjena frekvencije određenog tona za jednu oktavu, tj. prikazano je kako se frekvencija povečala dva puta. Nakon toga su prikazani svi korišteni alati (C#, WPF, WavePad, razne biblioteke), te kako sve ono što je objašnjeno sa strane digitalne obrade signala pretvoriti u C# kod.

LITERATURA

[1] http://blogs.zynaptiq.com/bernsee/dft-a-pied/#more-60, pristup ostvaren 1.9.2016.

[2] http://www.codeproject.com/Tips/1082074/Audio-Pitch-Shifter, pristup ostvaren 25.5.2016.

[3] http://blogs.zynaptiq.com/bernsee/pitch-shifting-using-the-ft/, pristup ostvaren 29.9.2015.

[4] http://mathworld.wolfram.com/FourierSeries.html, pristup ostvaren 26.9.2016.

[5] Lecture notes for Fourier transformation and its applications - Prof. Brad Osgood

[6] Head First C#, 3rd Eddition – Jennifer Greene, Andrew Stellman

[7] Professional WPF with C# and .NET – Christian Nagel

[8] https://msdn.microsoft.com/hr-hr/, pristup ostvaren 2.10.2015.

[9] https://msdn.microsoft.com/en-us/library/windows/desktop/dd371455(v=vs.85).aspx, pristup ostvaren 16.9.2016.

[10] https://cscore.codeplex.com, pristup ostvaren 26.5.2016

[11] http://www.nch.com.au/wavepad/fft.html, pristup ostvaren 26.9.2016.

SAŽETAK

Fourierova transformacija daje puno praktičnih alata za rastavljanje signala u jako puno pojedinačnih valova. Ti valovi su kosinusni ili sinusni valovi te sinusoide (opisane kombinacijom sinusa i kosinusa). Prednost istovremenog korištenja sinusa i kosinusa kod Fourierove transformacije je u tome da se otvara vrata uvođenju pojma faze koja transformaciju čini općenitijom na taj način da se Fourierova transformacija može koristiti učinkovitije i jasnije u slučajevima kada promatrani val nije sinusni niti kosinusni. Fourierova transformacija je nezavisna o signalu kojeg proučava u tom smislu da zahtjeva isti broj operacija bilo da se radi o jednostavnom ili kompliciranom valu. Ovo je razlog zašto se Diskretna Fourierova transformacija naziva bezparametarska transformacija, tj. nije direktno od pomoći kada je potrebna inteligentna obrada signala (u slučaju u kojem se zna da je signal koji se ispituje sinusoida , korisnije je dobiti informaciju o amplitudi, frekvenciji, i fazi nego jako puno sinusnih ikosinusnih valnih oblika na nekoj predefiniranoj frekvenciji).

Utvrđeno je da se ulazni signal procjenjuje na temelju mreže fiksnih frekvencija (binovi) za koje postoji mogućnost da neće imati nikakvu vezu sa frekvencijama ulaznog signala. Za mrežu koja se koristi pri analizi se može reći da je umjetna s obzirom da osoba koja radi analizu praktički sama bira sinusne i kosinusne valove s obzirom na njihovu učestalost. Zbog toga je odmah vidljivo da će osoba koja radi te analize gotovo sigurno naići na problem u kojem će frekvencije signala biti između frekvencija korisnikovih transformacijskih binova. Posljedica je ta da sinusoida koja ima frekvenciju koja je između dva bina neće biti dobro prikazana u transformaciji. Binovi koji su susjedni onome koji je najbliži frekvenciji ulaznog signala pokušati će popraviti to frekvencijsko odstupanje i zbog toga će energija ulaznog vala biti razvučena preko nekoliko susjednih binova. Ovo je također glavni razlog zašto Fourierova transformacija ne može tako lako analizirati zvuk i dati njegov osnovni harmonik i više harmonike (to je razlog zašto se sinusne i kosinusne valove naziva segmentima sinusoide, a ne harmonicima).

Na kraju se kada se napravio prikaz frekvencije tona A2 te prikaz tona A3, koji je dobiven transformacijom, jasno se pokazalo da se frekvencija dva puta povećala. To je pokazano još ranije u radu kada je objašnjeno da najniža frekvencija ovisi o rasponu između prvog i zadnjeg uzorka a najviša o rasponu između uzoraka, tj. što je više uzoraka između prvog i zadnjeg uzorka, frekvencija će biti veća.

ABSTRACT

Fourier transform gives a lot of practical tools for disassembly of signals in a lot of individual waves. These waves are cosine or sine waves and sinuous (described combination of sine and cosine). The advantage of simultaneous use of sine and cosine at the Fourier transform is that it opens the door introduction of the concept phase transformation makes it more general so that the Fourier transform can be used more effectively and more clearly in cases where the observed wave is not a sine nor cosine. Fourier transformation is independent of the signal, which studies in this respect that requires the same number of operations whether it be a simple or complicated wave. This is why the Discrete Fourier Transform called non-parametar transformation, ie. not directly helpful when you need an intelligent signal processing (in the case in which it is known that the signal under test sinusoid, useful to get information about the amplitude, frequency, and phase but a lot of sinus and cosine waveforms at a predefined frequency).

It was determined that the input signal is estimated based on a network of fixed frequencies (bins) for which there is a possibility that it will not have any connection with the frequency of the input signal. For the network to be used in the analysis can be said that artificial given that the person doing the analysis practically alone selects the sine and cosine waves according to their frequency. Therefore, it is immediately evident that the person doing this analysis almost certainly encounter a problem in which the signal frequency to be between frequency bins transformation member. The consequence is that the sine wave that has a frequency that is between the two stages will be well displayed in the transformation. Bins which are adjacent to the one that is closest to the frequency of the input signal to try to fix this frequency deviation and therefore the energy of the input waveform to be stretched over several neighboring bins. This is also the main reason why the Fourier transform can not easily analyze the sound and give its fundamental harmonic and higher harmonics (that's why the sine and cosine waves called partials, but no harmonics)

At the end when the tone frequencies A2 and A3 were displayed, which is obtained by the transformation, it is clearly shown that the frequency increased twice. It was shown earlier in the work when it is explained that the lowest frequency depends on the range between the first and the last sample and the maximum of the range of samples, ie. As the samples between the first and the last sample, the frequency will be greater.

ŽIVOTOPIS

Danijel Dželajlija rođen je 14. Travnja 1991. u Osijeku. Pohađao je osnovnu školu „Retfala“. Nakon toga upisuje Elektrotehničku i prometnu školu u Osijeku. Potom upisuje preddiplomski studij elektrotehnike koji uspješno završava u roku. Nakon toga slijedi diplomski studij na smjeru elektroenergetike. Zadnju godinu diplomskog upisuje apsolventsku godinu i počinje raditi u tvrtki Phoenix d.o.o. gdje počinje raditi kao programer u programskim jezicima *Visual basic* i *C#* te se koristi grafičkim podprogramima *Windows Forms* i *Windows Presentation Foundation.* U istoj tvrtki počinje raditi i sa KUKA robotskim rukama, SMC koračnim motorima, Lenze i Beckhoff komunikacijskim uređajima i korisničkim sučeljima. Od stranih jezika poznaje jedino engleski jezik koji tečno govori i razumije.