Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Pávlicz Ádám Boldizsár

VORBIS AUDIOTÖMÖRÍTŐ ELJÁRÁS OBJEKTÍV ÉS SZUBJEKTÍV VIZSGÁLATA

KONZULENS

Dr. Rucz Péter

BUDAPEST, 2018

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Pávlicz Ádám Boldizsár**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2018. 12. 06.

...…….
Pávlicz Ádám Boldizsár

Összefoglaló

Szakdolgozatomban bemutatom az emberi hallás azon jellemzőit, amelyeket kihasználva veszteségesen, de érezhető minőségromlás nélkül tudunk tömöríteni hanganyagokat. Bemutatom a Vorbis audiotömörítő eljárás működését, a hanganyagok kódolásának és dekódolásának folyamatát. A dekódolás folyamatát Matlab és Simulink szoftverek segítségével megvizsgálom. Bemutatom a PEAQ modellt, és ennek segítségével megvizsgálom, hogy a Vorbis különböző bitrátájú tömörítések során mekkora érehető minőségromlást okoz. Összeállítottam egy szubjektív tesztet, ebben a tesztalanyok különböző minőségű veszteséges hangmintákat értékeltek a veszteségmentes referenciajelhez képest. A teszt lebonyolításához írtam egy programot Java nyelven, és egy MySQL adatbázist használtam. A tesztek eredményét Matlab segítségével elemeztem ki. Legvégül összehasonlítom a PEAQ modell eredményeit a szubjektív teszt eredményeivel.

Abstract

In my thesis I explain auditory masking, and how we can use it in lossy audio compression to produce smaller audio files, while the lossy audio sample remains indistinguishable from the lossless reference sample for the average person. I show the workings of the Vorbis audio codec, with an emphasis on audio encoding and decoding. I use Mathworks' MATLAB and Simulink to examine the process of decoding. With the help of the PEAQ model, I compare the quality of Vorbis' compression on different bitrates. I compiled a test, in which participants were asked to rate multiple lossy samples compared to a reference lossless sample. I implemented said test in JAVA and used a MySQL database to store the results. I analyzed the results using Mathworks' MATLAB. Lastly I compare the results from the PEAQ model and from my test.

1 Bevezetés

1.1 Motiváció

Szeretek zenét hallgatni, mindig is szerettem. Azon kívül, hogy fontos nekem hogy mindig szóljon valami, az is fontos hogy jó minőségű legyen. A témaválasztásban főleg ez motivált. Ahogy egyre idősebb lettem, és egyre több stílust, egyre több előadót ismertem és szerettem meg, azzal találtam szembe magamat, hogy nem férnek rá egyszerre a telefonomra. Először vettem bele nagyobb memóriakártyát, de ez is csak egy rövidtávú megoldás volt. A helyzeten az javított nagyon sokat, hogy elkezdtem különböző veszteséges formátumokat kipróbálni különböző minőségben. Eleinte nagyon alacsony bitrátákat adtam meg, mert ezzel így nagyon sok helyet lehetett megtakarítani, viszont gyorsan feltűnt, hogy így a minőség nagyon sokat romlott.

1.2 A dolgozat célja

A szakdolgozatom fő célja az volt, hogy számomra – és az átlagos ember – számára találjak egy olyan beállítást, ami optimális minőség és méret szempontjából is, de egyértelműen a minőség megtartása volt a fontosabb e kettő közül. Dolgozatomban a veszteséges audiotömörítéshez kapcsolódó fogalmak, mint például a maszkolás és az alapvető határok körüljárása után a Vorbis audiotömörítő eljárást szeretném bemutatni és vizsgálni. Ezt különböző objektív és szubjektív tesztek elvégzésével tettem, először ismertetve a teszteket és a hozzájuk felhasznált eszközöket, technológiákat, majd pedig az ezek által adott eredményeket elemezve külön-külön és együtt is. A Vorbis vizsgálata során a téma nagysága miatt nem a teljességre törekedtem, hanem azokra a részekre fektettem a hangsúlyt, amelyek a hangminőséget befolyásolják.

1.3 Köszönetnyílvánítás

Köszönetet szeretnék mondani Dr. Rucz Péter tanár úrnak, a segítsége és iránymutatása nélkül ez a szakdolgozat nem készülhetett volna el.

Szeretném megköszönni mindenkinek, aki végighallgatta az általam összeállított tesztet, és ezzel segítette a munkámat.

2 Veszteséges audiotömörítés

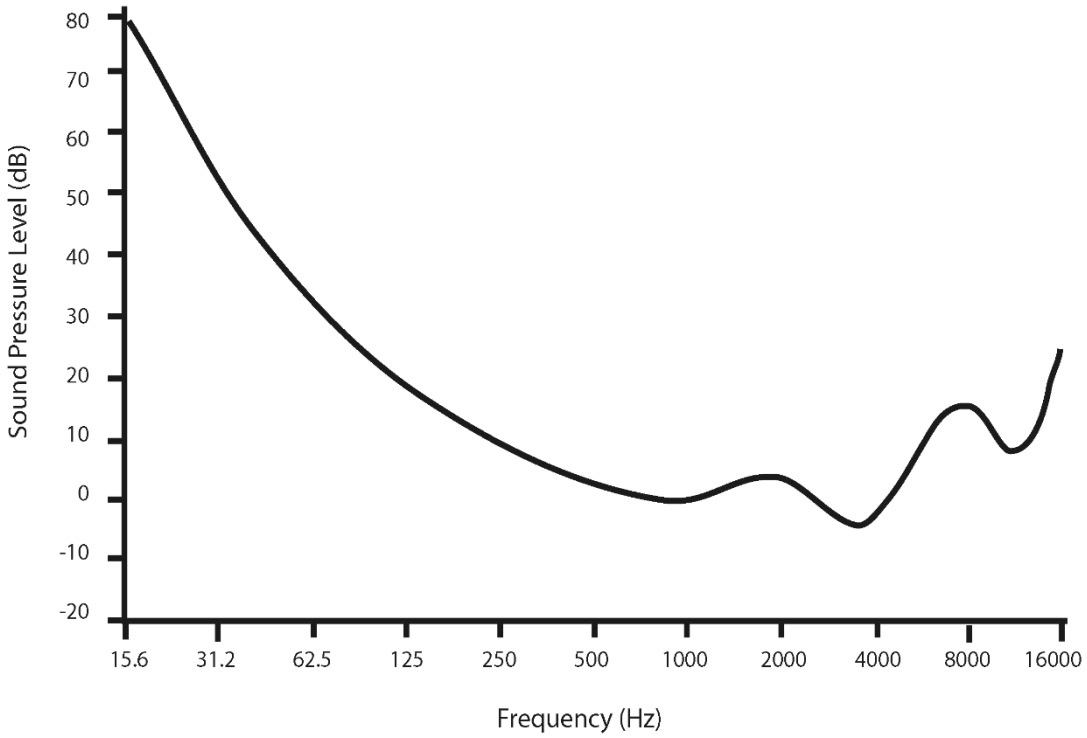
Napjainkban szinte mindenki fogyaszt valamilyen audiotartalmat, legyen az zene utazás közben, hangoskönyv alvás előtt vagy akár a kedvenc podcastünk. Minél jelentősebb része életünknek ez, annál fontosabb, hogy ezt minél hatékonyabban tegyük. Ha például utazás közben szeretnénk zenét hallgatni, akkor jó hogyha minél több szám elfér a véges tárhelyű lejátszónkon, vagy pedig minél kevesebb mobilinternetet használunk el hozzá. Ebben tud nekünk segíteni a veszteséges audiotömörítés, melynek segítségével egy hanganyagot úgy tárolunk el, hogy a tömörítés ne okozzon érzeti különbséget, de mégis rengeteg helyet vagy küldendő adatot spóroljunk meg. Ahhoz, hogy a tömörítés valóban ne okozza az észlelt hangminőség romlását, pszichoakusztikai modellt használunk a tömörítés során. Az ilyen modellek biztosítják, hogy a hanganyagból csak azokat a részeket hagyjuk el, amiket az emberi fül már nem képes érzékelni.

2.1 Pszichoakusztikai alapok

A veszteséges audiotömörítéshez alapvetően az emberi hallás két tulajdonságát használjuk ki, a hallásküszöböt és a maszkolást.

2.1.1 A hallásküszöb

Az emberi hallás elméletileg a 20 Hz és 20 kHz közötti frekvenciákat képes érzékelni, de ez csak egy közelítő érték, a tényleges tartomány emberről emberre változik, viszont audiotömörítés szempontjából ezek jó átlagos értékek. A maximális észlelhető rezgéstartomány az életkor előrehaladtával csökken, de sok mástól is károsodhat a hallásunk. A másik fontos tulajdonsága a hallásunknak, hogy különböző frekvenciákon eltérő érzékenységű. Az 1. ábrán látható, hogy milyen frekvencián (vízszintes tengely, Hz) milyen érzékeny (függőleges tengely, dBSPL) a hallásunk. Egy adott frekvenciához minél magasabb érték pont tartozik, annál kevésbé érzékeny rá a fülünk. [3]



1. Ábra: Átlagos emberi hallásküszöb

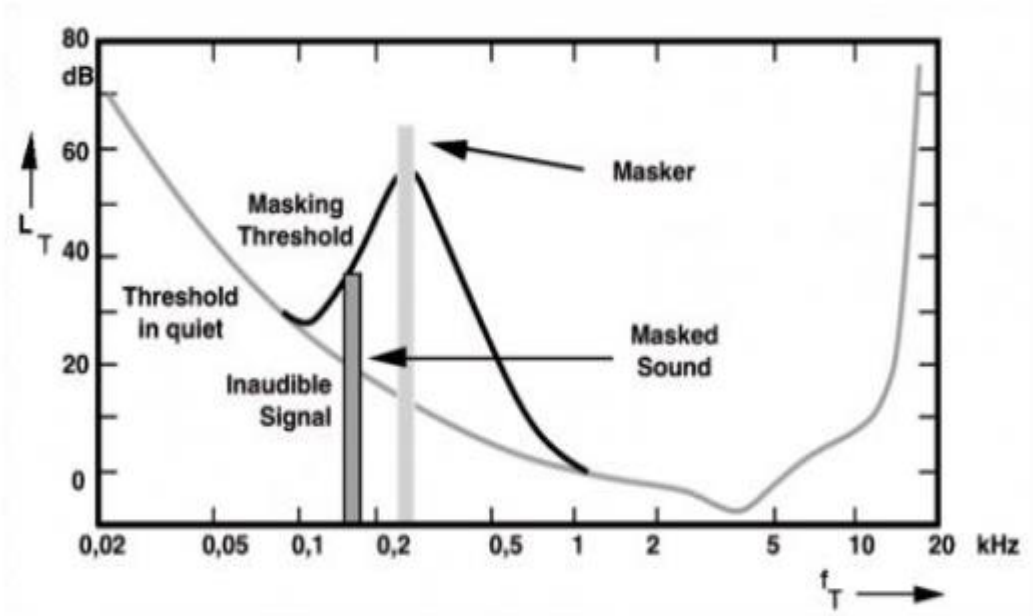
Az, hogy milyen érzékeny valamilyen hangra a fülünk függ a jel frekvenciájától, és attól is, hogy a jel milyen típusú: atonális vagy tonális. Az atonális hangokhoz nem tudunk egy érzett hangmagasságot társítani, mint például valamilyen szélessávú zaj (ha az FM rádiót olyan frekvenciára állítjuk, amin nincs adás akkor ilyet hallhatunk), a tonális hangokhoz viszont tudunk egy viszonylag jól meghatározható hangmagasságot társítani, ilyen például egy zongorán egy billentyű leütése után megszólaló hang. Ezeket a típusokat a hallásunk következő tulajdonsága miatt kell megkülönböztetnünk, ami a maszkolás. Természetesen egy összetett hangban egyszerre jelen vannak atonális és tonális komponensek, ezért mindkettővel foglalkoznunk kell a tömörítés során.

2.1.2 Maszkolás

Abban az esetben, ha a sok beérkező hangból nem mindegyiket érzékeltük, maszkolás történt. Ha két egyidőben érkező hangról van szó, akkor frekvenciatartománybeli maszkolás, ha különböző, de egymáshoz közeli időpontban érkező hangokról akkor időtartománybeli maszkolás. Például, ha éppen egy koncerten vagyunk, nem valószínű, hogy meghalljuk a telefonunk csörgését, vagy egy tűzijáték robbanás után nem halljuk pár pillanatig a mindig jelenlévő városi alapzajt.

2.1.2.1 Frekvenciatartománybeli maszkolás

Ez a jelenség a csiga (szerv) felépítéséből adódik. A beérkező rezgés (hang) a csigában meghatározott részt rezget meg az amplitúdójával arányos mértékben. Ez alapján a csiga a hallóidegen keresztül egy jelzést küld az agynak, hogy milyen frekvenciájú hang érkezett. Azonban ez a meghatározott rész nem pontszerű, hanem egy kisebb terület. Ez azért fontos mert így, ha egyszerre két, viszonylag közeli frekvenciájú hang érkezik, azok egymáshoz nagyon közeli területeket rezgetnek meg, így az erősebb hang miatt nem biztos, hogy érzékeljük a gyenge hang által keltett rezgést. Itt fontos, hogy a beérkező hang tonális vagy atonális, mert a különböző típusok különböző mértékben maszkolják ki a körülöttük lévő frekvenciákat. [4]



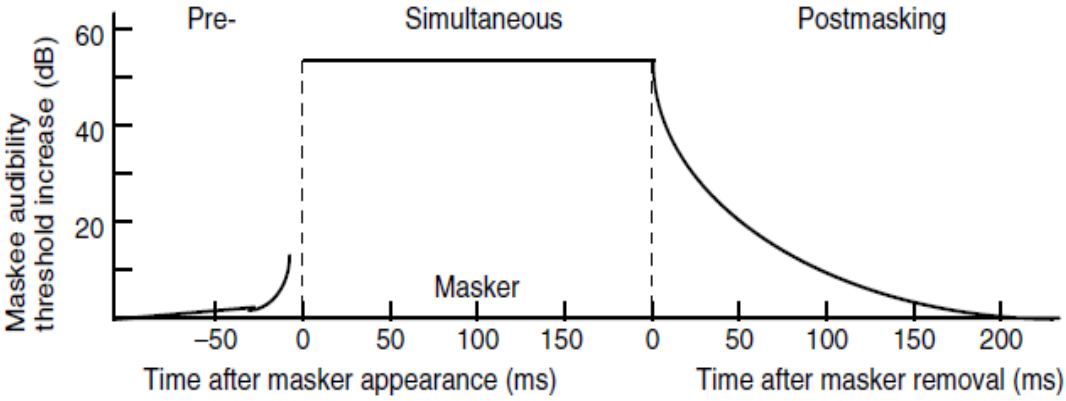
2. Ábra: Frekvenciatartománybeli maszkolás

A maszkoló hang maga körül tulajdonképpen megemeli a hallásküszöböt (2. ábra), és a vele egyidőben érkező, de az újonnan létrejött hallásküszöb alá eső hangokat egyszerűen nem érzékeljük. Ezt a tulajdonságot nagyon jól ki tudjuk használni a veszteséges tömörítés során.

2.1.2.2 Időtartománybeli maszkolás

Egy hang nem csak a vele egyidőben érkezőket tudja kimaszkolni, hanem az előtte vagy az utána érkezőket is. A veszteséges tömörítés során általában a forward maskingot használjuk ki, ami egy hangosabb hang utáni halkabb hang maszkolását jelenti, mert ennek az ideje 100-120 ms is lehet, ami meghaladja a tömörítés során használt időszeletek

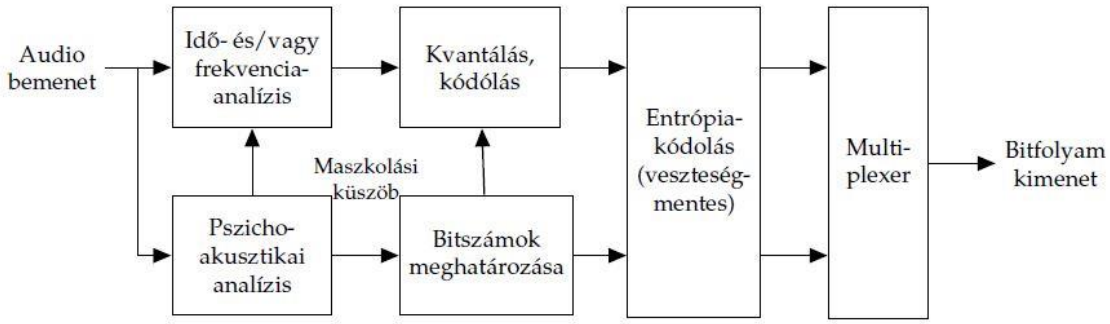
méretét. A backward maskingot, amikor egy hangos hang az előtte levő halkabbat maszkolja ki, nem használjuk, mert ennek az ideje mindössze 2–5 ms körüli. A backward masking magyarázata, hogy a hangosabb hangok által keltett ingerület az idegpályákon gyorsabban halad, és így megelőzheti az előtte lévő halkabb hang jelét.



3. Ábra Időtartománybeli maszkolás

3 Ogg-vorbis

A 4. ábrán látható, hogy egy általános veszteséges audiotömörítőnek milyen részfeladatokat kell megvalósítania.



4. Ábra: Veszteséges audiotömörítő eljárások általános blokkvázlata

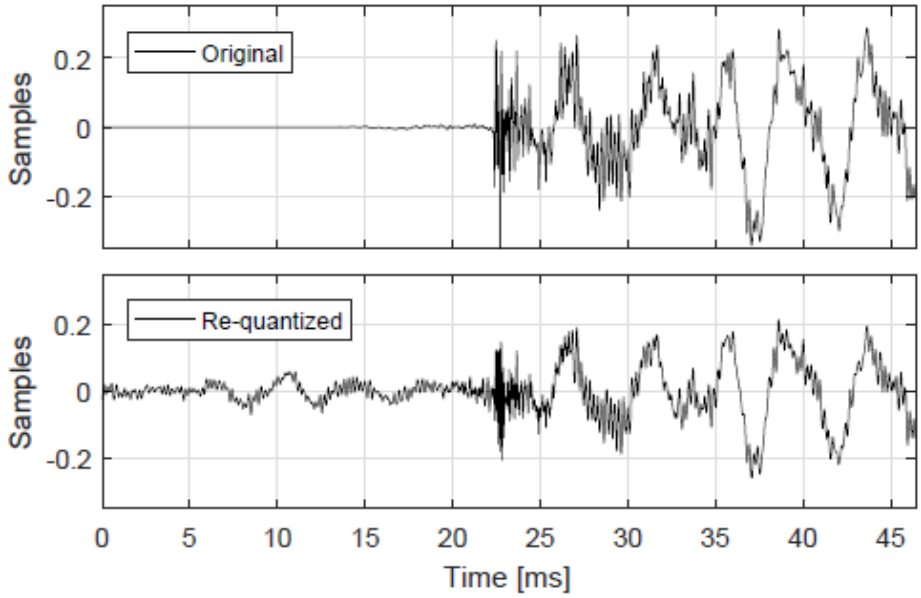
A szakdolgozatomban főleg a Vorbis konkrét működésével foglalkozom. A vorbis egy nyílt forráskódú, veszteséges audioformátum, amelyet a Xiph.org Foundation fejlesztett. A Vorbishoz szükséges egy konténer formátum is, ami leggyakoribb esetben az Ogg, amit szintén a Xiph.org Foundation fejlesztett, de mivel nyílt forráskódú, ezért akár sajátot is írhatunk hozzá. A Vorbis codec egy általános célú, alapvetően változó bitrátájú audiotömörítő. [1]

Minden Vorbis audiofájl elején található néhány kötelező fejlécmező, melyek:

- Identification header: Ez azonosítja a bitfolyamot Vorbisként, továbbá itt található a mintavételezési frekvencia, a Vorbis verziója, csatornák száma és hasonló egyszerű karakterisztikák.
- Comment header: Itt találhatóak a metaadtok (tagek), például az előadó, a szám címe, és az album címe, ezen kívül az előállító enkódernek/programnak egy azonosítója.
- Setup header: Ebben található minden, ami a CODEC beállításához szükséges dekódolásnál, többek között az összes VQ (Vector quantization) és Huffman kódkönyv is.

3.1 Vorbis audiotömörítés lépései

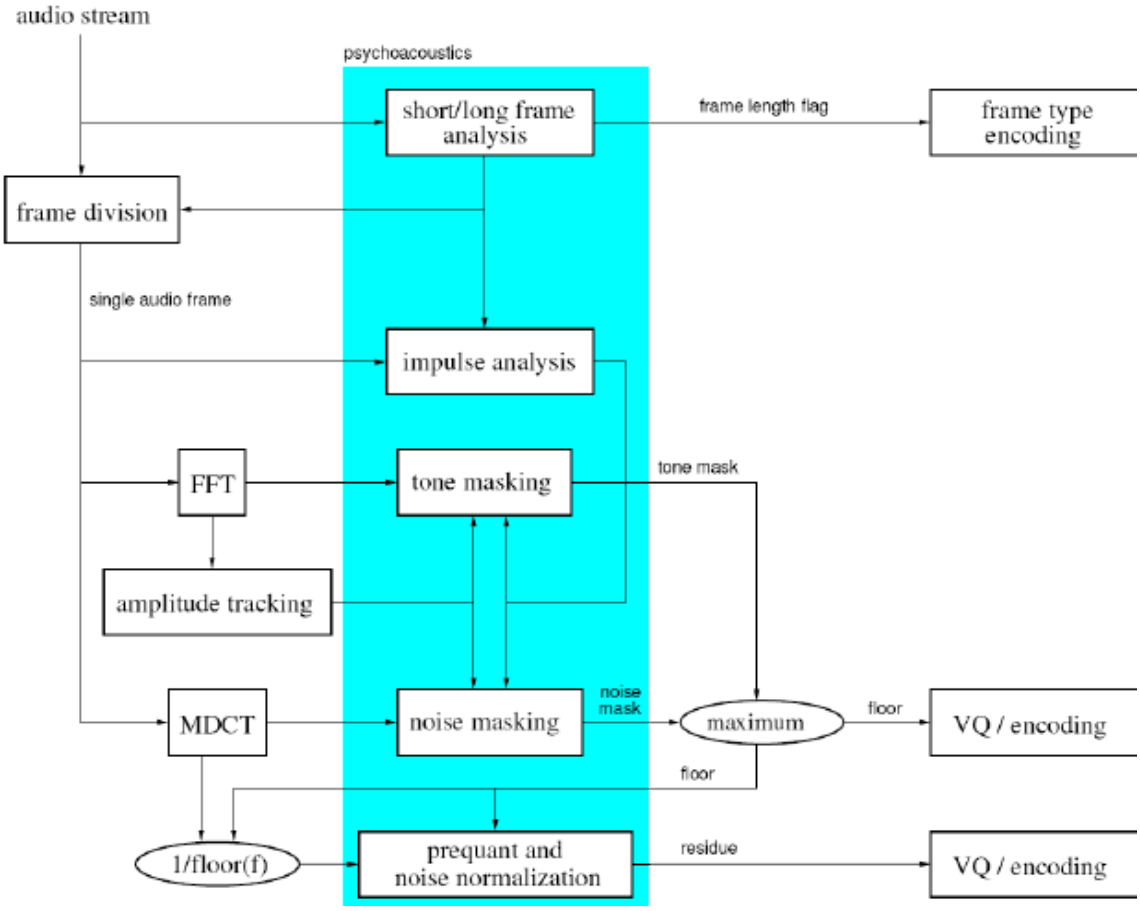
A bemenő audiojel feldolgozása blokkokban történik. A Vorbis rövid és hosszú ablakokkal dolgozik, amikre azért van szükség mert ezekkel csökkenthető a kvantálási zaj. Ahogy a 5. ábrán is látszik, ha egy hosszú ablakban van egy hosszabb szünet vagy halk rész, akkor, ha az egész ablakot kvantáljuk, a zaj ezen a részen is megjelenik. Az első lépés tehát, hogy eldönti a kódoló, hogy az adott bemenő jelfolyam kis vagy nagy blokkokkal kódolandó-e. Ennek eldöntését a pszichoakusztikai modell végzi, majd ezután a kódoló beállítja a blokkhossz jelző mezőt (flag-et) az adott blokk fejlécében.



5. Ábra Kvantálási zaj szétterülése a teljes ablakban [3]

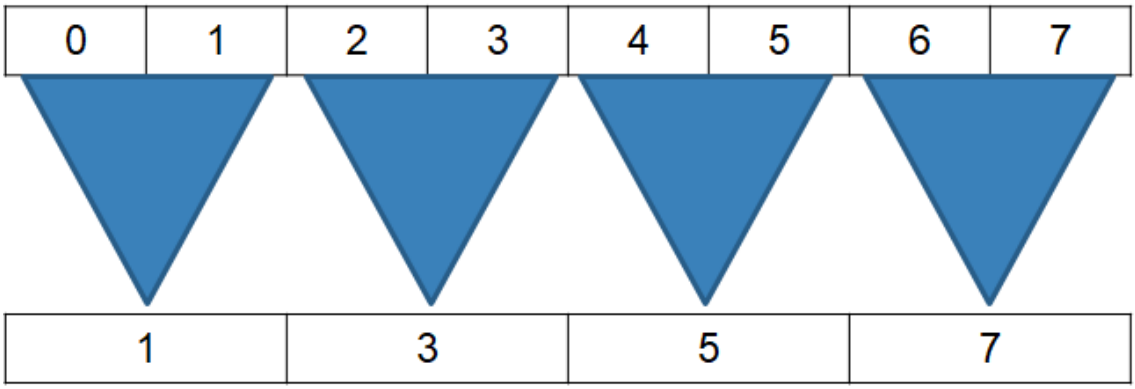
Miután a bitfolyamból kivettünk egy megfelelő hosszúságú blokkot, következhet ennek a szeletnek (frame-nek) a kódolása (6. ábra). A Vorbis enkóder FFT segítségével határozza meg a tonális komponenseket, és ezek maszkolását, és MDCT (Modified discrete cosine transform) segítségével az atonális komponensek maszkolását. Ezek után egy maximumot számol ezekből, és az emberi hallásküszöbből, és ez alapján állítja össze – a minőségi beállításnak megfelelően – az ún. floort. A floor egy alacsony felbontású reprezentációja az audiospektrumnak a framen belül, az adott csatornán. Miután a floor megvan, az MDCT eredményét és a floort felhasználva előállítja az ún. residue-t is. A residue a frame audiospektrumának a finoman felbontott része, miután a floor-t kivontuk belőle. Dekódolásnál ennek a két vektornak (floor és residue) a

skalárszorzatából (elemenkénti szorzatából) állítható vissza majd (veszteségekkel) az audiojel. A floor és a residue előállításához is a pszichoakusztikai modellt használtuk.



6. Ábra: Vorbis tömörítés folyamata [2]

Az így létrejött floor és residue vektorokat ezek után egy vektor kvantálóval (vector quantization, 7. ábra) tömörítjük. Ez egy veszteséges eljárás, ami úgy működik, hogy az egymáshoz közel lévő értékeket, előre meghatározott, hozzájuk legközelebb álló vektorhoz rendeli.



7. Ábra: Egyszerű vektor kvantálás példa

A CODEC rengeteg kódkönyvet tárol, amiből a kódoló kiválasztja a neki legmegfelelőbbet mind a floorhoz, mind a residuehoz, majd a frame fejlécében eltárolja hogy melyiket használta, mert ez szükséges a dekódoláshoz. A vektor kvantálás után még történik egy veszteségmentes Huffman-kódolás, amivel még több helyet takaríthatunk meg.

A Vorbis alapvetően egy változó bitrátájú audiotömörítő, ez a gyakorlatban úgy valósul meg, hogy amikor elindítjuk a kódolást egy jelsorozatra akkor többféleképpen felparaméterezhetjük ezt a kérést, attól függően, milyen minőségű és/vagy méretű fájlt szeretnénk a végén. Az egyik módja ennek, hogy megadunk neki egy minimális és egy maximális bitrátát (kbit/s), és a kódoló a bejövő adat függvényében eldönti, hogy egy adott blokkhoz ő e között a két érték között mit használjon. Ha a maximum és a minimum nagyon közel van egymáshoz, akkor tudunk konstans bitrátájú fájt generálni, de ez nem célszerű, mivel így rengeteg optimalizációs lehetőséget veszünk el a kódolótól. Egy másik mód a bitráta beállítására az, hogy megadunk neki egy cél bitrátát, és ő ennek az értéknek a közelében lévő bitrátákat használ az adott blokkokra. Ezt a célbitrátát megadhatjuk a beépített -q (quality) kapcsolóval is, ez -q0 és -q10 között változhat a Xiph.org által fejlesztett Vorbisban, de különböző harmadik fél által továbbfejlesztett kódolókban előfordulnak ennél kisebb értékek is. Az aoTuVb3 kódoló például támogat -q-2 (~32 kbit/s) és -q-1 (~48 kbit/s) kapcsolóállást. [8]

A Vorbis többféle lehetőséget ad arra, hogy a különböző csatornákat hogyan kezeljük. A sztereó információt négyzetes poláris leképezéssel kezeli, ami akkor nagyon hasznos, ha a bal és jobb csatorna között nagy a korreláció. Ez természetesen nem csak két csatornára működhet, a hivatalos Vorbis implementáció 255 csatornát támogat. A Vorbisban minden egyes csatornának a spektruma normalizálva van egy padló függvényhez képest (floor function), ami a csatorna spektrumának egy alacsony felbontású képe. A négyzetes poláris leképezésnél (square polar mapping) a sztereó fázis egy adott frekvenciakomponensnél a két csatorna közötti normalizált amplitúdó különbségét jelöli. Ha a sztereó információk már rendelkezésünkre állnak mint amplitúdó és sztereó fázis, ezeket az alábbi módon tudjuk tárolni.

A jelenlegi referencia kódolóban lehetőségünk van veszteségmentes (lossless), és point stereo csatolásra, illetve ezek keverékére (mixed stereo). A veszteségmentes sztereó ekvivalens azzal, mintha a két csatornát egymástól függetlenül kódolnánk, de mivel a csatornák residue vektorait fűzzük össze, így helyet takaríthatunk meg az egymástól

független kódoláshoz képest. A point stereo módszernél a csatornák sztereó fázisát teljesen eldobjuk, ez hozzájárul a tömörítő veszteségességéhez, itt minden sztereó információ a két csatorna spektrális padlójának (spectral floor) a különbségéből származik.

A csatornák kezelésének módját is paraméterként adhatjuk meg, de a különböző minőségi beállításokhoz tartoznak alapértelmezett csatornakezelések, ha ezt mi külön nem specifikáljuk.

Kapcsoló	Cél bitráta (kbps)	Bitráta tartomány (kbps)	Csatorna kezelés
-q0	64	64 – 80	point / lossless
-q1	80	80 – 96	point / lossless
-q2	96	96 – 112	point / lossless
-q3	112	112 – 128	point / lossless
-q4	128	128 – 160	point / lossless
-q5	160	160 – 192	point / lossless
-q6	192	192 – 224	lossless
-q7	224	224 – 256	lossless
-q8	256	256 – 320	lossless
-q9	320	320 – 500	lossless
-q10	500	500 – 1000	lossless

A Vorbis -q6 beállítás felett már csak veszteségmentes csatornakapcsolást használ, ami érthető is, hiszen ezek már nagyon jó minőségű hanganyagok, amiknél valószínű, hogy fontosabb a minőség, mint a helymegtakarítás. A táblázatban a -q5 azért van külön kijelölve, mert ez az a tömörítési beállítás, ami felett az átlag felhasználó transzparensnek érzékeli a tömörített anyagot (ha az eredeti jelfolyam veszteségmentes volt), azaz e felett nehéz, a referencia meghallgatása nélkül pedig szinte lehetetlen megkülönböztetni a veszteségmentes és a tömörítésből visszaállított hanganyagot.

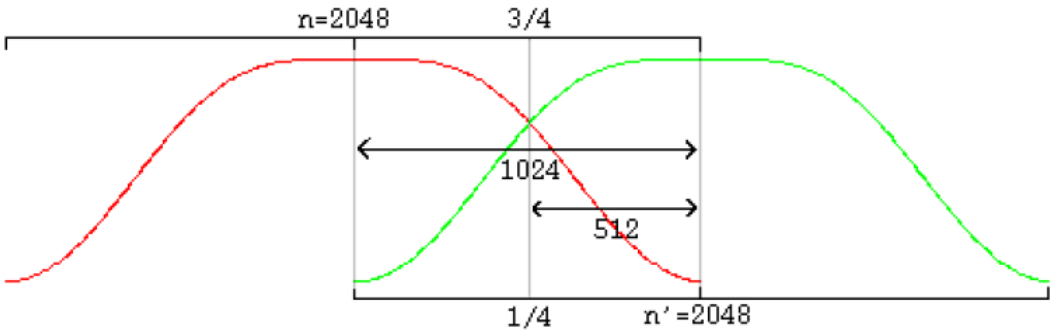
3.2 Ablakfüggvények és átfedések a blokkok között

A bejövő jelfolyam feldarabolása blokkokra és ezek külön-külön kódolása más-más ablakmérettel és bitrátával jelentős mennyiségű helymegtakarítást tesz lehetővé. Azzal, hogy egy blokkon belül csökkentjük a bitszámot, kvantálási zajt viszünk a jelbe.

Amíg a kvantálási zajt a hangmintánk kimaszkolja, addig ez nem okoz a blokkon belül problémát, viszont a különböző blokkokban különböző mennyiségű kvantálási zajunk van, és ez a blokkok határán ugrásszerű változásokat okoz, amik számunkra is hallhatóak lennének. Azért, hogy ezeknek az ugrásszerű változásoknak a hatását csökkentsük, ablakfüggvényeket használunk, és a blokkokat átfedésbe hozzuk. A Vorbis egy a szabványban meghatározott ablakfüggvényt használ: [1]

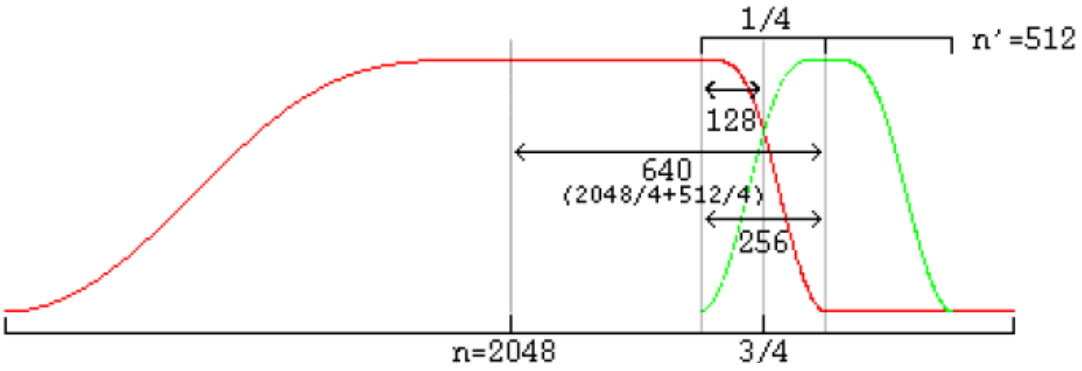
$$y = \sin(0.5 \cdot \pi \cdot \sin^2(\frac{x + 0.5}{n \cdot \pi})$$

A szabvány azt is előírja, hogyan kell átfedni az egymás után következő blokkokat. Ha két azonos hosszúságú blokk követi egymást (8. ábra) akkor az első ablak felétől kezdődik a második ablak, így 50%-os átfedés lesz a kettő között.



8. Ábra: Azonos hosszúságú ablakok átfedése

Ha azonban két különböző hosszúságú ablak követi egymást, ahogy a 9. ábrán is látható, akkor a hosszabb ablak alakját át kell alakítanunk, hogy zökkenőmentesen váltson egyik ablak a másikba. A dekódolónak szabad számon tartania, hogy az előző és a következő ablak milyen hosszú a mostanihoz képest, de egy blokkon belül két jelzőmező (flag) is a segítségére van, amik kódoláskor kerültek bele a blokk fejlécébe. Ezek meghatározzák az előtte és utána következő blokk ablakméretét. Ez azért lehetséges mindössze két egybites jelzővel, mert egy keret csak kétfajta ablakméretet használhat, egy rövidet és egy hosszút, viszont ezeket a kódolás elején csatornánként szabadon megválaszthatja 64 és 8192 bit között, annyi megkötéssel, hogy az ablakméretnek kettő hatványának kell lennie.



9. Ábra: Két különböző ablak közötti átlapolás

4 Vorbis vizsgálata Matlab és Simulink segítségével

4.1 Próbajelek előállítása

Ahhoz, hogy megvizsgáljam mennyit romlik vagy hogy romlik-e egyáltalán a minősége egy hanganyagnak egy adott -q beállításnál, először egyszerű tesztjeleket állítottam elő Matlab segítségével. Az első ilyen jel egy 1 kHz-es négyszögjel, wav formátumban. Ezután Audacity program segítségével ebből a wav fileból különböző minőségű ogg-vorbis fájlokat hoztam létre, ezeket fogom összehasonlítani az eredeti jellel idő- és frekvenciatartományban.

A négyszögjel létrehozása után készítettem egy 50 Hz-es fűrészjelet is, szintén Matlab segítségével. Ezután ebből szintén Audacity segítségével hoztam létre a különböző minőségű ogg-vorbis hangmintákat.

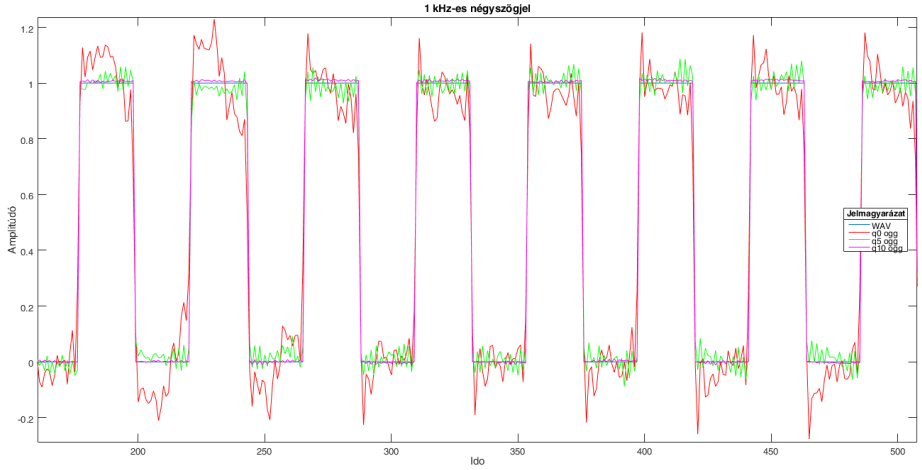
4.2 Próbajelek vizsgálata időtartományban

A négyszögjelet háromféle beállítással ábrázoltam, -q0, -q5, -q10, mert ezek jól lefedik a különböző alkalmazási területeket. A -q0 egy alacsony bitrátájú (~64 kbit/s) ogg-vorbis kódolás eredménye, ez azokban az esetekben lehet hasznos, ha kis sávszélességű jelet akarunk kódolni, vagy ha sokkal fontosabb számunkra a kis fájlméret, mint a minőség (de természetesen még mindig felismerhető az eredeti hangminta). A -q5-ös beállítás egy közepes bitrátájú kódolást eredményez (~ 160 kbit/s). Ez a minőség az átlagembernek már elég ahhoz, hogy ne tudja megkülönböztetni érdemben a veszteségmentes és a tömörített hanganyagot, viszont a veszteségmenteshez képest jelentős helymegtakarítás érhető el. Végül a -q10-es beállítás a legmagasabb, amit a hivatalos Vorbis specifikáció támogat, ez ~500 kbit/s -es bitrátát jelent. Ennél a beállításnál egyértelműen a minőség a fontos, talán abszolút hallással lehet csak megkülönböztetni (sajnos ezt nem volt lehetőségem tesztelni), de még az átlagosnál jobb hallással rendelkezők sem fogják érezni a különbséget a veszteségmenteshez képest. Azért éri meg a használata mégis, mert ugyan minőségben majdnem azonos a veszteségmentes hangmintákhoz képest, sokkal kevesebb helyet foglal el a lemezünkön, ezt a 10. táblázatban is láthatjuk.

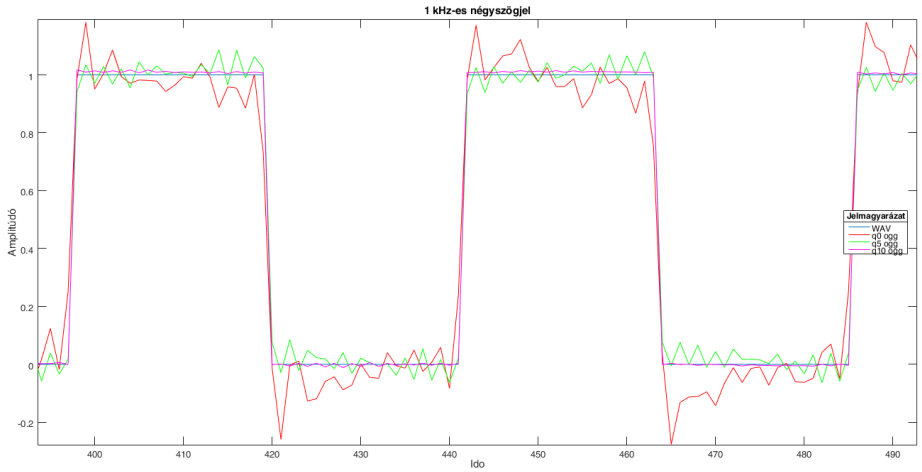
Kódolás	veszteségmentes	-q0	-q5	-q10
Fájlméret	862 KB	59 KB	148 KB	268 KB

10. Táblázat: tízmásodperces, 1 kHz-es négyszögjel mérete

A négy fájlt ezután (az eredeti wav, és a három különböző ogg) beolvastam Matlabban és ábrázoltam őket, hogy látszódjon mennyivel torzult a tömörített jel az eredetihez képest.



11. Ábra: 1 kHz-es négyszögjel különböző tömörítéssekkel

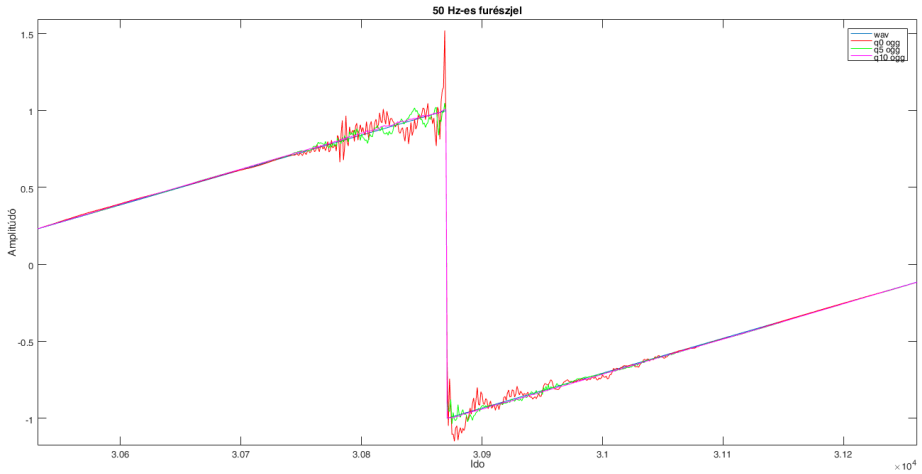


12. Ábra: 1 kHz-es négyszögjel különböző tömörítésekkel, belenagyítva

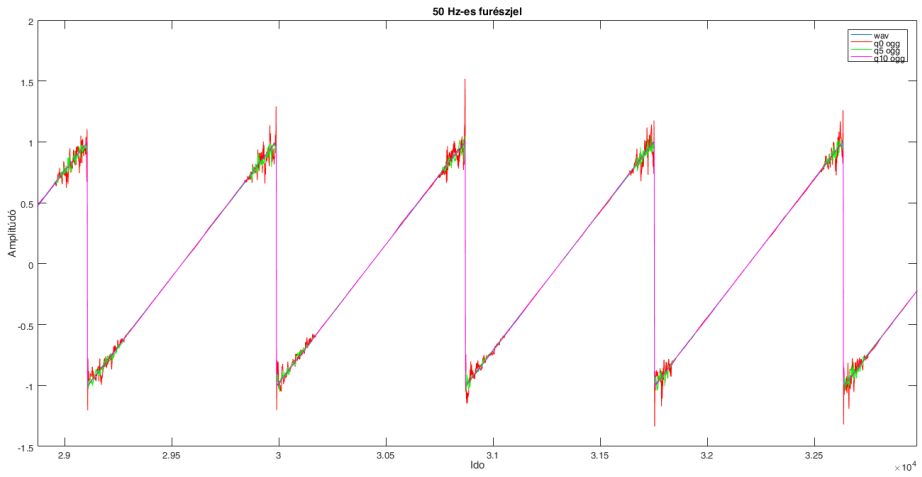
A 11. és 12. ábrán a veszteségmentes WAV fájl kék színnel lászik. Ehhez képest a magenta színű -q10-es ogg-vorbis fájl vonala mindössze pár százalékkal tér el, de nincsen jelentős túl- és alullövés. A zöld színű jel a -q5-ös ogg-vorbishoz tartozik, itt

körülbelül 10% a maximum eltérés az eredeti jelhez képest, viszont körülbelül egy szinten marad a négyszögjel vízszintes részein. A -q0-ás jelhez a piros színű jel tartozik, itt jelentős, akár 20%-os eltérés van a jelben az eredetihez képest, és a kezdeti túllövés után folyamatosan csökkenő tendenciát mutat a következő váltásig.

Ugyanezeket a lépéseket elvégeztem az 50 Hz-es fűrészjellel is, hogy ott is hasonló eredmények jönnek-e ki, és a 13. és a 14. ábrán látszik, hogy igen, sőt a -q0-ás kódolásnál majdnem 25%-os eltérések is előfordulnak a veszteségmentes és a veszteségesen tömörített hanganyag között.



13. Ábra: 50 Hz-es fűrészjel egy ugrása különböző tömörítésekkel

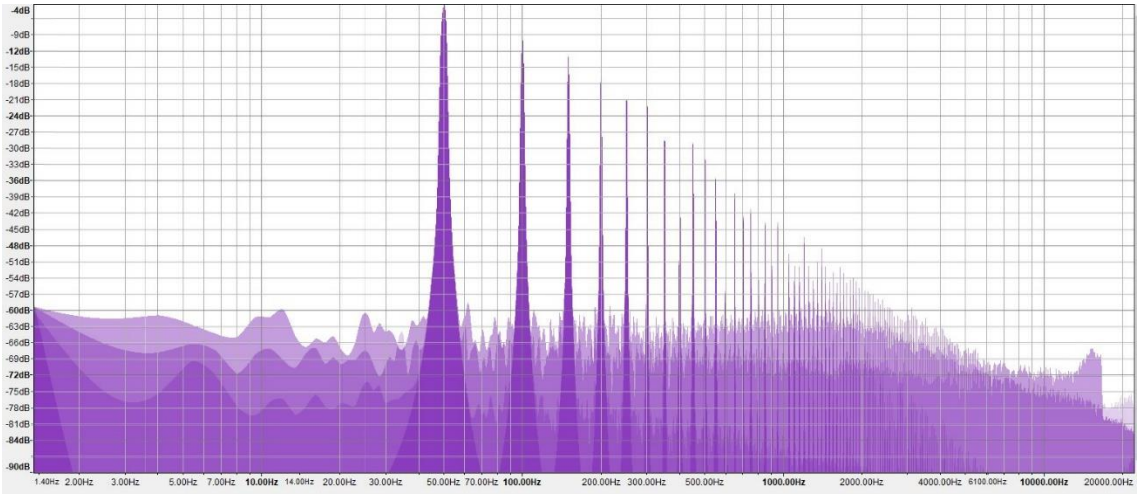


14. Ábra: 50 Hz-es fűrészjel időtartományban különböző tömörítésekkel

4.3 Próbajelek vizsgálata frekvenciatartományban

Az időtartománybeli elemzés után a két próbajel spektrumának változását vizsgáltam különböző -q minőség beállítások mellett. Nagyobb tömörítés hatására megnő a kvantálási zaj, az alacsony minőségű tömörítésnél annyira, hogy már hallhatóvá válik.

A jelek spektrumát az Audacity program spektrumanalizátorával készítettem, Hanning ablakkal és 32768-as mintaszámmal, logaritmikus frekvenciatengely skálázással. Ezután képként lementettem őket egyenként és Photoshop segítségével egymásra vágtam őket. Ahhoz, hogy a Photoshop használata ne okozzon torzítást, a tengelyeket azonosan állítottam be, mind a 4 esetben.



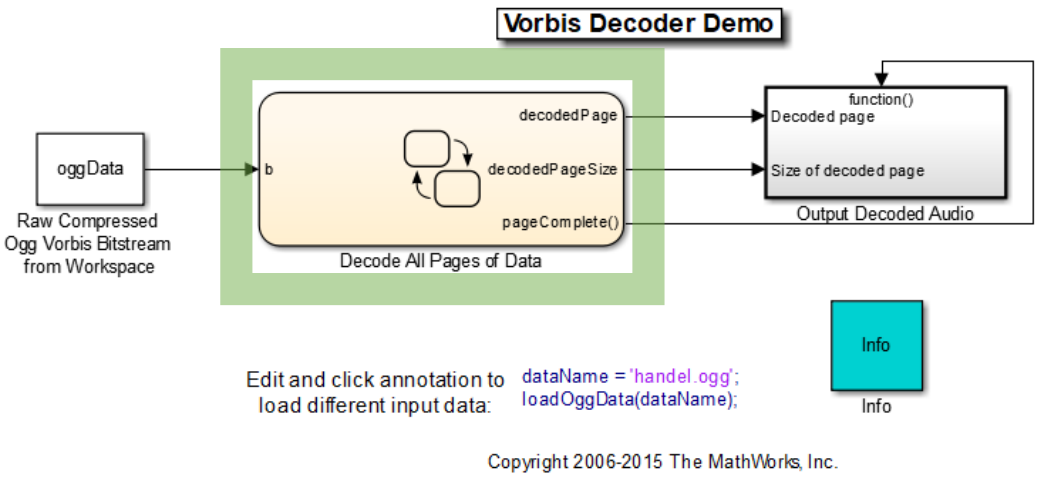
15. Ábra: 50 Hz-es fűrészjel frekvenciaspektruma különböző minőségű tömörítésekkel

A ábrán a legsötétebb lila színnel a veszteségmentes fűrészjel spektruma látszódik, jól kivehető rajta az 50 Hz-es alapharmonikus és a felharmonikusok az 50 Hz minden egész számú többszörösénél. A világosabb lila szín a 15. ábrán, kisebb bitsűrűségű kódoláshoz tartozik. Érdemes megfigyelni az 50 Hz alatti tartományt, itt elvileg nem kellene semmilyen spektrális sűrűségnek lennie, viszont a tömörítés során a bitszámok csökkentése miatt bekerülő kvantálási zaj, és a maszkolási görbe megemelkedése miatt itt is megjelennek frekvenciakomponensek. Ezek még a legrosszabb minőségű tömörítésnél is 56 dB-lel kisebbek, mint az alapharmonikus amplitúdója, ami még mindig olyan jel-zaj viszont eredményez, ami mellett felismerhető az eredeti jel, de már átlagos emberi füllel is érezhető minőségromlás történik. A -q0 beállításnál megfigyelhető az is, hogy a kevesebb bitszám, és megemelt maszkolási görbe mellett még úgy is tömörít, hogy a 16 kHz feletti frekvenciakomponenseket egyszerűen

elhagyja, itt látható is egy körülbelül -15 dB-es ugrás. Amíg a veszteségmentes hangfájl frekvenciakomponensei körülbelül 30 Hz és 4200 Hz közé, a legjobb minőségű tömörített hangfájl frekvenciakomponensei 0 Hz és 6100 Hz közé esnek. A közepes és a legrosszabb minőségű hangagyagok összetevői viszont 0 Hz és 22050 Hz közé esnek a megemelkedett maszkolási görbe miatt, így jóval több kvantálási zajt visznek a mintába, ennek ellenére sokkal kevesebb helyet foglalnak, mert ezeket a viszonylag azonos amplitúdójú részeket nagyon jól lehet vektor kvantálással és aztán Huffman-kódolással tömöríteni.

4.4 Vorbis vizsgálata Simulinkkel és Matlabbal

A Simulink a MathWorks (a MATLAB fejlesztője) által fejlesztett program, amit modellezésre, szimulálásra és grafikus programozásra használhatunk. A Simulink a MATLAB-bal szorosan integrálva van, onnan meg is hívhatjuk, illetve a Simulink is képes MATLAB szkripteket indítani. A Simulinknek lassan megszámlálhatatlanul sok kiegészítő modulja van, én a szakdolgozatomban az alap Simulink mellett az Audio System Toolbox-ot, ezen belül pedig konkrétan az Audio Processing Algorithm Design-on belül található Vorbis Decodert használtam. Ezt legegyszerűbben MATLAB-ból indíthatjuk az „audiovorbisdecoder” paranccsal. [5]



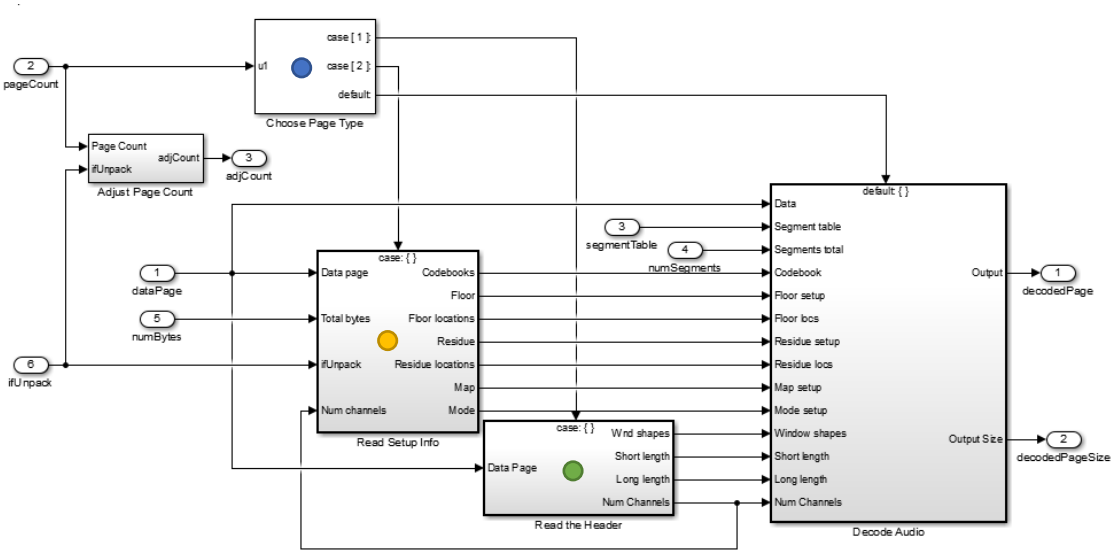
16. Ábra: Simulink Vorbis Decoder kezdőlapja

Miután elindítottuk ezt a Simulink programot, a dataName annotáción keresztül adhatunk meg neki egy ogg-vorbis fáljt, amit dekódolni szeretnénk. A 16. ábrán látható baloldali doboz azért felelős, hogy a munkaterületről megkapja a Simulink az adatot, a középső doboz pedig azért, hogy ezt feldolgozza és továbbadja a jobboldali doboznak, amelyiknek már csak annyi a dolga, hogy szintetizálja a hangot a bejövő adatok

függvényében. Én a középső dobozzal foglalkoztam, ugyanis ebben található meg a Vorbis dekódoló.

4.4.1 A dekódolás menete

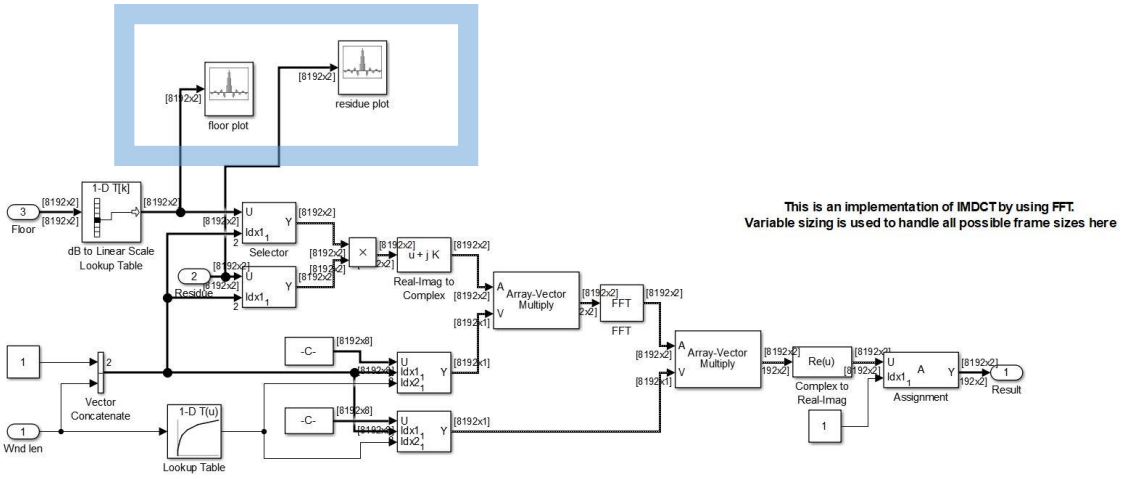
Ha kicsit beleássuk magunkat a modell szerkezetébe, könnyebben megértjük a dekódolás menetét. Első lépésként az Ogg konténerformátumból oldalanként ki kell nyernünk a Vorbis bitfolyamot. Mivel az ogg oldalanként tárolja a bitfolyamot ezért oldalanként is dolgozzuk fel, de ha valami más konténerformátumunk lenne, vagy valahonnan egy Vorbis bitfolyamot kapnánk, akkor nem lenne szükségünk erre. Természetesen az Ogg oldalankénti tárolása nem csak egy felesleges adalék, a Vorbis nem rendelkezik saját szinkronizációval és hibavédelemmel, ezeket a konténerformátum biztosítja.



17. Ábra: Egy oldal dekódolása Simulinkben

Az első két lap különösen fontos (17. ábrán a kék pontos doboz), ezért ezt külön figyeljük, ezekben találhatóak a hangminta visszaállításához elengedhetetlen fejlécek. Az első fejlécben található a csatornák száma, a rövid és hosszú ablakok mérete, továbbá itt (17. ábrán a zöld pontos doboz) előállítjuk a rövid és hosszú ablakokhoz tartozó ablakfüggvényeket. A második oldalt a 17. ábrán sárga ponttal jelölt doboz fogja feldolgozni, ez az oldal tárolja a dekódoláshoz szükséges összes kódkönyvet, előfordulható floor, és residue vektorokat. Az ezután következő oldalak már audioblokkokat tartalmaznak. Ezeket az egyszerűbb számításokat Simulinkben végezve, a bonyolultabb számolásokat pedig Matlab scriptek segítségével dekódolja. Az

audioblokkból kivesz egy csomagot, ebben ellenőrzi egy csomag azonosító 1 bites flag (packet type) értékét, ha a ez egyenlő 0-val akkor egy audiocsomaggal van dolgunk, úgyhogy mehet tovább a dekódolás. A csomag fejlécéből és a kódkönyvekből származó adatok alapján egy matlab script segítségével kiszámolja a floor és residue vektorokat, majd ezeket továbbadja egy IMDCT (Inverse Modified Discrete Cosine Transform) blokknak. Ez annak a transzformációnak az inverze, amit kódoláskor használtunk. A Simulink Vorbis Decoder ezt FFT segítségével valósítja meg.



18. Ábra: IMDCT implementáció kiegészítve

Az IMDCT-t implementáló blokkon belül szerettem volna vizsgálni a floor és a residue vektorokat különböző minőségben, ezért ide a 18. ábrán kék keretben látható plottereket kötöttem rá a megfelelő adatbuszokra. Ehhez az 1 kHz-es négyszögjelet használtam.

A floor plotteren látható egy 1 kHz-es -q0 minőséggel kódolt négyszögjel floor vektora. Jól látszott rajta, hogy ez valóban a spektrum egy alacsony felbontású „képe”. Ehhez tartozik egy residue vektor is, ami a residue plotteren látható. A végső jel ennek a két vektornak az elemenkénti szorzatából fog összeállni. A floor vektor értékei körülbelül 140 decibelnyi értéket vehetnek fel (körölbelül 24 bit előjel nélkül), és az audiospektrum vektor (a floor és a residue elemenkénti szorzata) legalább 120 decibeles értéket kell, hogy felvegyen a specifikáció szerint (körülbelül 21 bit előjellel). Ezért, ha a floor -140 decibelen van, a residue vektornak 0 dB és +140 decibel közötti értéket kell felvennie, hogy le tudja fedni az egész tartományt, ha viszont a floor 0 decibelre van „odaszögezve” akkor a residue vektornak 0 dB és -140 dB közötti értéket kell felvenni ehhez. Ebből a

két szélsőséges esetből következik, hogy a residue vektornak egy körülbelül 280 decibeles tartományt kell lefednie, -140 dB és +140 dB között, amihez közelítőleg 48 bitre van szüksége, előjellel együtt. Tehát a floor és residue elemenkénti szorzatából összeálló audiospektrum vektornak egy $24\,\text{bit} \cdot 48\,\text{bit}$-es szorzás eredményét kell kezelnie, emiatt érdemes legalább 64 bitesre választani, még ha csak egy 16 bitet kezelő lejátszó eszköznek adjuk is tovább a dekódolt jelet.

Mielőtt eljutna egy hangszóróig a jel, még hátra van az egyik legfontosabb dolog, az átlapolás. Erre a különböző ablakok szélein jelentkező ugrások miatt van szükség. Innentől már csak pár apróbb simítás és a kimeneti tárolóba (pufferbe) kerül az adat, ahonnan a Simulink továbbküldi a rendszerünk alapértelmezett audiokimenetére. Ahhoz, hogy megbizonyosodjak róla, hogy a kapott floorból és residueból tényleg elő lehet állítani a jelet, még a kimenet elé bekötöttem egy spektrumanalizátor blokkot. Jól látszott rajta, hogy ez valóban egy négyszögjel spektruma, az 1 kHz-es alapharmonikus után minden páratlan számú egész többszörösnél jelentek meg a felharmonikusok.

4.5 PEAQ algoritmus

A Perceptual Evaluation of Audio Quality egy standardizált algoritmus az érzékelt hangminőség objektív megállapítására. 1994 és 1998 között fejlesztették ki szakértők az akkori Nemzetközi Távíró és Telefon (technikai) Konzultatív Bizottság Rádiókommunikációs szektorának (ITU-R) egyik munkacsoportjából. Az eljárás lényege, hogy szoftver segítségével szimulálja az emberi fül tulajdonságait, és a különböző modellek kimeneteit egy 1 és 5 közötti számmal értékeli. Ez a szám megmondja, hogy mennyivel érzékeli rosszabbnak egy átlagos ember az adott hangmintát egy referencia hangmintához képest. [6]

ITU-R veszteségi skála	Jelentése
5.0	észrevehetetlen
4.0	észrevehető, de nem idegesítő
3.0	kicsit idegesítő
2.0	idegesítő
1.0	nagyon idegesítő

4.5.1 PEAQ algoritmus Matlab script segítségével

A Matlabban megírt PEAQ modell 0 és -4 közötti értékeket ad vissza mint ODG (objektív különbség értékelés), ami egyszerű eltolással megfeleltethető az ITU-R veszteségi skálának. A Vorbis kódolót hétféle hangmintával, ezeknek különböző minőségű kódolásával teszteltem. [9]

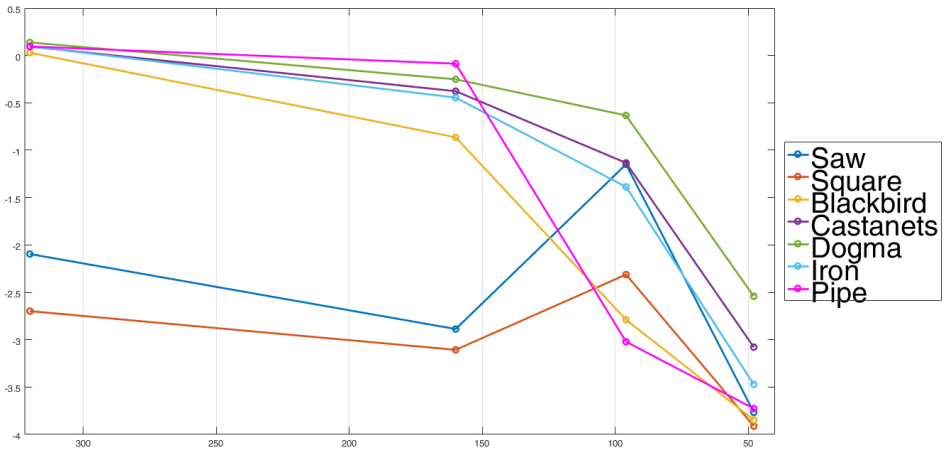
A hangminták:

- 1 kHz-es négyszögjel
- 50 Hz-es fűrészjel
- Castanets
- Pipe
- Blackbird
- Iron
- Dogma

A négyszögjelet és a fűrészjelet azért használom itt is, mert már rendelkezésre állnak, és kíváncsi voltam ezekre is működik-e ez a modell. A Castanets hangminta nagyon érzékeny a pre-echora (kvantálási zaj szétterülése az időtartományban), mert rengeteg csendes – hangos rész követi hirtelen egymást. A Pipe nevű hangmintában skótdudák szólalnak meg, ebben a hangmintában nagyon erős tonális komponensek vannak, érzékeny a zajra. A Blackbird-öt azért választottam, mert ebben egyszerre van jelentősége a tonális (zongora) és az atonális (maraca) komponenseknek. Az Iron a leghosszabb minta a választottak közül (34 másodperc), ebben van egyedül énekhang a választott minták közül, ami szerintem fontos része a vizsgálatnak. Legvégül a Dogmát azért választottam, mert ez eredetileg is egy nagyon zajos, torzított minta, kíváncsi voltam ezzel hogyan birkózik meg a Vorbis.

Ezeket a kiválasztott jeleket négyféle célbitrátával teszteltem, ezek 320 kbps, 160 kbps, 96 kbps, és 48 kbps. Azért esett ezekre a bitrátákra a választásom, mert a Spotify is majdnem ezeket használja, annyi a különbség, hogy az ő szolgáltatásukban elérhető legalacsonyabb minőség körülbelül 24 kbps, de a hivatalos Vorbis implementációnak nincs ennyire alacsony -q kapcsolóállása.

4.5.1.1 Az eredmények



19. Ábra: ODG pontszámok ábrázolva

A 19. ábra alapján lászik, hogy a PEAQ modellt nem ilyen egyszerű tesztjelekre találták ki, mint a fűrész és a négyszögjel, ezeknél a viselkedése nem teljesen kiszámítható, de azért hagytam benne őket a minták között, hogy ez kiderüljön. Az algoritmus szerint a 96 kbit/s bitráta adja vissza a legjobb minőségű (ezek közül) fűrész- és négyszögjelet, de majd az 5. fejezetben lévő hallgatásos tesztből kiderül, hogy az átlagos emberi fül nem így ítéli meg. A másik öt jelnél nagyon érdemes megfigyelni a 320 kbit/s oszlopot az alábbi táblázatban, ez az algoritmus szerint szinte megkülönböztethetetlen az eredeti referenciajeltől az összes rendes mintánál, stílustól függetlenül.

PEAQ modell által adott ODG pontszámok

	320 kbit/s	160 kbit/s	96 kbit/s	48 kbit/s
Fűrészjel	-2,09	-2,88	-1,14	-3.76
Négyszögjel	-2,69	-3,10	-2,31	-3.91
Blackbird	0,02	-0,86	-2,78	-3,84
Castanets	0,09	-0,37	-1,13	-3,07
Dogma	0,13	-0,25	-0,63	-2,54
Iron	0,09	-0,44	-1,38	-3,47
Pipe	0,09	-0,08	-3,02	-3,72

Ami ehhez még nagyban hozzátartozik, itt viszont nem került bele az adatok közé, hogy mennyivel kevesebb helyet foglal így a hangfálj, mint veszteségmentesen. A Blackbird.wav 1048 KB, a Blackbird320.ogg pedig csak 226 KB. Így gyakorlatilag minőségvesztés nélkül majdnem az ötödére csökkentettük a szükséges tárhelyet vagy adatforgalmat. Ha ennél jóval kisebb (fele akkora) célbitrátával kódoljuk az adatokat, már talán érdemes elkezdeni rá figyelni, hogy milyen típusú mintát szeretnénk kódolni, de ennél a bitrátánál legrosszabb esetben is csak észrevehető a különbség a referencia jelhez képest, de még nem idegesítő. Így, hogy lejjebb adtunk a minőségből, még több helyet tudunk megtakarítani. Az előző esetben közel ötödére tudtuk csökkenteni a hangfájlunk méretét, viszont 160 kbit/s célbitrátával már csak 114 KB lett a Blackbird hangfájlunk, ami azt jelenti, hogy egy alig észrevehető minőségbeli romlással több mint 88% helyet spóroltunk. Ennél a célbitrátánál csak alapvetően zajos, torzított jeleknél érdemes lejjebb menni, ha azt szeretnénk, hogy ne tűnjön fel a veszteség. A Dogma még 96 kbit/s bitrátával is szinte megkülönböztethetetlen a referenciamintától, és míg az eredeti veszteségmentes Dogma.wav 1036 KB-ot foglal, addig ez, a Spotify normál hangminőségnek megfelelő bitrátájú Dogma96.ogg csak 69 KB-ot. Ez több mint 93%-os hely- és/vagy adatforgalom-megtakarítás. Ezalatt a célbitráta alatt viszont hirtelen nagyon gyorsan elkezd romlani a minőség és szinte hallgathatatlan lesz a tömörített audiotartalom.

5 Szubjektív vizsgálat

A PEAQ modell egy objektív értékelést adott arra, hogy egy veszteségmentes referenciajelhez képest, mennyivel érezheti rosszabbnak egy átlagos ember a veszteségesen tömörített hangmintát. Kíváncsi voltam, hogy ha én is meghallgattatom ezeket a mintákat több emberrel, akkor vajon hasonló eredmények jönnek-e ki, vagy esetleg pont, hogy valamilyen nagy ellentmondás keletkezik a PEAQ modell eredménye és a hallgatásos teszt eredményei között.

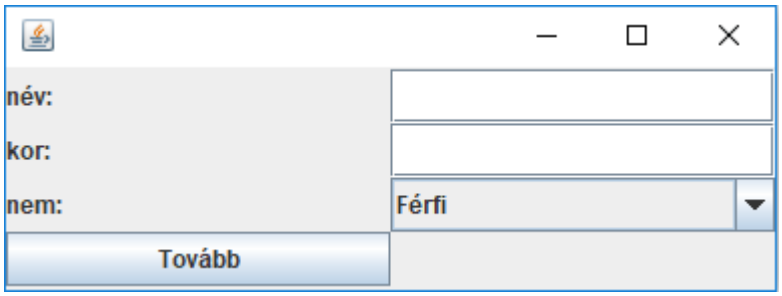
5.1 Előkészületek

A tesztet minden résztvevővel ugyanazzal a fejhallgatóval hallgattattam meg, hogy a különböző minőségű megszólaltató eszközök közötti különbségek ne befolyásolják az eredményt. Ez a fejhallgató egy Audio-Technica M40fs típusú precíziós stúdió fejhallgató volt. Két okból esett erre a fejhallgatóra a választásom, egyrészt mert a hangszórói képesek lefedni az egész emberi hallás frekvenciatartományát jó minőségben, másrészt azért, mert ez volt elérhető.

A meghallgatásos teszthez írtam egy programot Java nyelven, és ennek eredményeit egy MySQL adatbázisba mentettem.

5.1.1 A meghallgatásos teszt menete

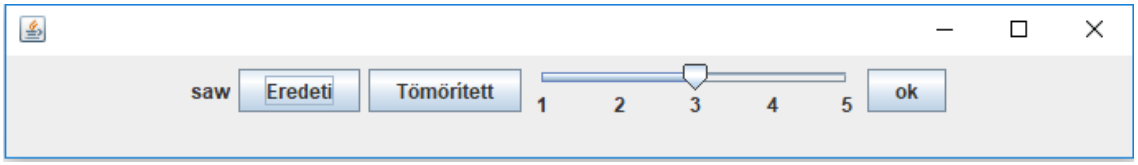
A teszthez nyolcféle mintát használtam, ebből hét ugyanaz, mint a PEAQ modellnél használt hangminták, azért, hogy az eredményeket könnyen össze lehessen hasonlítani. Ezen kívül minden hallgató választhatott egy számára kedves zenéből kivágott (maximum 15–20 másodperces) részletet.



20. Az adatok megadására szolgáló ablak a teszt elején

A teszt elején a hallgató megadja a nevét, életkorát, és nemét a 20. ábrán látható módon. A nevet azért tárolom el, hogy a teszt befejeztével, ha érdekli az alanyt akkor vissza tudja nézni, mit hogyan értékelt. Az életkort és a nemet azért mentem el, hátha ezeknek van valami hatása arra, hogy az adott alany hogyan értékelte a különböző minőségű mintákat.

Az adatok megadása után a hallgató kiválasztja azt a már előre elkészített veszteségmentes hangmintát, amire ő kíváncsi különböző minőségben. Ezt a hangmintát a program átkonvertálja különböző minőségű ogg-vorbis hangfájlokká. Arról, hogy mindezt hogyan teszi, a programról szóló részben írok.



21. Ábra: Az értékelés felülete

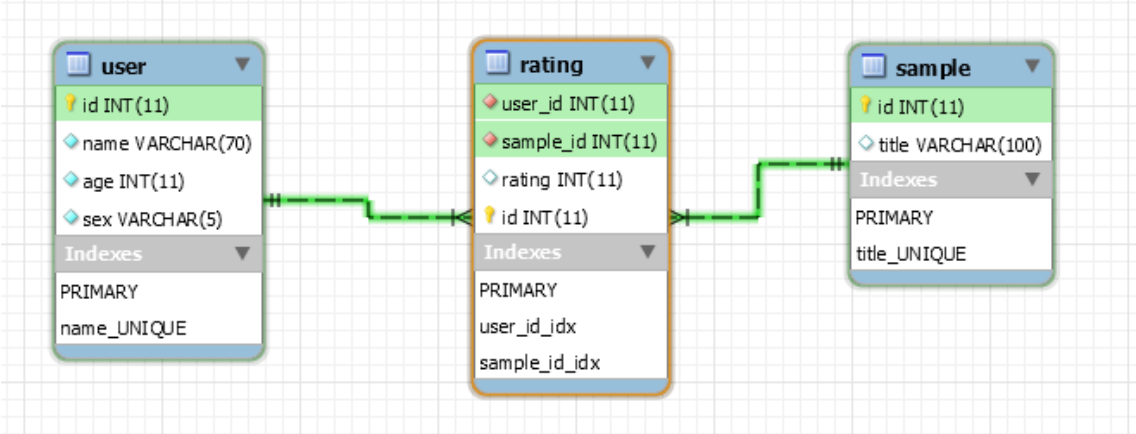
A konvertálás végeztével a hallgatók előtt megjelenik egy új ablak, benne azzal a felülettel, amin keresztül meghallgathatják a hangmintákat és értékelhetik őket (21. ábra). A hangminták csoportjai sorban követik egymást a következő sorrendben:

1. 50 Hz-es fűrészjel
2. 1 kHz-es négyszögjel
3. Blackbird
4. Castanets
5. Iron
6. Pipes
7. Dogma
8. A saját hangminta

Minden egyes csoporton belül 5 darab mintát kell értékelniük a hallgatóknak, ezek mind különböző minőségűek. Az öt minőség legrosszabbtól a legjobbig: Veszteségmentes (wav), 320 kbit/s vorbis, 160 kbit/s vorbis, 96 kbit/s vorbis, 48 kbit/s vorbis. A hallgató nem látja, hogy éppen milyen minőségű hangmintát hallgat, csak azt, hogy ez melyik csoport éppen. A tesztalanyok az Eredeti gomb megnyomásával tudják meghallgatni a veszteségmentes referenciajelet, a tömörített gomb megnyomásával pedig az éppen aktuális (még nem értékelt) tömörített jelet. A teszt előtt ők úgy tudják, hogy öt darab, különböző minőségű veszteségesen tömörített hanganyagot fognak értékelni

csoportonként, de ez hazugság. Valójában négy veszteségesen tömörített hanganyagot értékelnek csoportonként, és egyszer a referenciajelet is összehasonlítják saját magával. Ezt azért így állítottam össze, mert kíváncsi voltam, hogy a hallható különbségek mellett befolyásolja-e az értékelést az, hogy úgy tudják a hallgatók, hogy veszteséges hanganyagot hallgatnak. A referenciamintát is és az aktuális mintát többször is meghallgathatják egymás után, majd a csúszka segítségével tudják értékelni. Az értékelésre 1–5 skálát használok, és a hallgatóknak előtte elmagyaráztam a különböző számok jelentését az ITU-R veszteségi skála alapján. Az „ok” gomb lenyomásával az értékelést a program elmenti, és betöltődik a következő minta, vagy ha a következő csoport jön, akkor annak a csoportnak a referenciajele és a következő minta. A csoportok ugyan sorban jönnek egymás után, de egy csoporton belül véletlenszerűen következnek egymás után a különböző minőségű értékelendő minták. Egy ilyen teszt elvégzése körülbelül 20–25 percet vesz igénybe.

5.1.2 A teszthez írt program és adatbázis



22. Ábra: UML diagram

Az adatok tárolására egy MySQL szervert használtam. A szerveren létrehoztam egy szakdolgozat nevű sémát, ebben hoztam létre a táblákat. Az adatok tárolására három táblát használok, ezek a user, rating, és sample táblák (22. ábra). A user táblában tárolom a tesztben résztvevő alanyok nevét, életkorát és nemét, továbbá tartozik hozzá id nevű elsődleges kulcs, amit az adatbázis generál. A sample nevű táblában találhatóak a rekordok nevei, és hozzá egy – az adatbázis által generált – egyedi azonosítószám. A rating táblában találhatóak az értékelések, melynek adatai a user_id, ami egy felhasználót azonosít, a sample_id, ami egy hangmintát azonosít, és egy egész szám, ami a felhasználó által adott értékelés a sample_id-nak megfelelő hangmintára. Itt felvettem

még egy külön id-t is az értékelésnek, ha esetleg egy ember többször is ki szeretné értékelni a hangmintát, akkor lehetősége legyen rá.

A programot, ami a teszt lebonyolítását végzi, Java nyelven írtam. A grafikus felület egyszerű Java.awt és Swing elemekből épül fel. A teszt során a felhasználó a saját hangmintáját egy grafikus fájlválasztó segítségével adhatja hozzá.

Miután a hallgató kiválasztotta a fáljt, a program a JAVE2 (Java Audio Video Encoder 2) könyvtár segítségével konvertálja azt. A JAVE2 könyvtár egy wrapper (burkoló) könyvtár az FFmpeg projectre, hogy java nyelven könnyen tudjunk egy formátumból egy másikra átkódolni. Ha sikeres az átkódolás, akkor megjelenik az értékelő ablak. Itt az „Eredeti” vagy a „Tömörített” gombra kattintva, egy új szálon elindul a lejátszás. A lejátszáshoz a javax.sound.sampled könyvtárat használom, ami a nyelv része. Miután a hallgató a csúszkán beállította az értékelését és rákattintott az „ok” gombra, a program a java.sql API-n keresztül küld egy INSERT kérést a MySQL szervernek, így mentve az eredményt. A szerver jelenleg a localhost 3306-as portján fut, így a program is itt próbál hozzá csatlakozni. A program sorrendben betölti a megfelelő mintacsoportokat, de ezen belül véletlen sorrendben értékelteti ki a hallgatóval az egyes mintákat. [10]

5.2 Eredmények

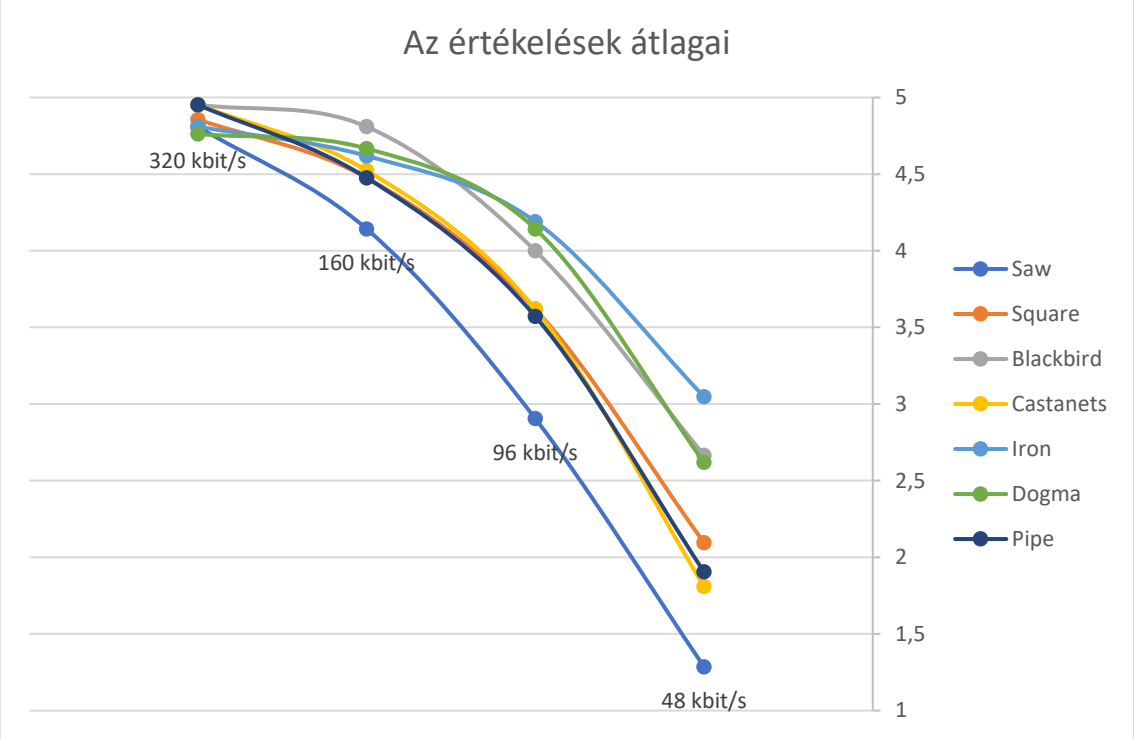
A tesztet 21 emberrel sikerült kitöltetnem, ez sajnos kicsit kevesebb, mint a célként kitűzött 25–30, de logisztikai okok miatt ennél több tesztet nem tudtam megszervezni.

5.2.1 A Veszteségmentes hangminták értékelései, saját magukhoz hasonlítva

Minta neve	Értékelések Átlaga	Értékelések szórása
50 Hz-es Fűrészjel	4.80	0.39
1 kHz-es Négyszögjel	4.76	0.68
Blackbird	5.00	0.00
Castanets	4.85	0.34
Iron	4.52	0.58
Pipes	4.85	0.34
Dogma	4.80	0.39

Ezekből az eredményekből látszik, hogy az is befolyásolja azt, hogy mennyivel érzünk rosszabbnak valamit egy másikhoz képest, hogy milyen előítélettel állunk hozzá a vizsgálatához. Az értékelők úgy tudták, hogy olyan mintákat értékelnek, amiket veszteségesen tömörítettünk, emiatt olyan előítéleteik születhettek, hogy ezeknek rosszabbnak kell lennie az eredeti hangnál. A két különös eset a Blackbird és az Iron. Az előbbinél mindenki eltalálta, hogy ez az eredeti, vagy legalábbis, hogy megkülönböztethetetlen az eredetitől, az utóbbi pedig majdnem egy fél értékkel rosszabbat kapott, mint kellett volna neki. Az Iron esetében valószínűleg közrejátszik, hogy ez a leghosszabb minta a maga 34 másodpercével, és ez – tapasztalatból mondva – elég arra, hogy ne emlékezzünk tisztán, milyen volt az eredeti jelünk, amihez hasonlítjuk. Ha ehhez még hozzávesszük, hogy azt mondták nekünk, hogy ez veszteséges, úgy tűnik, hajlandóbbak vagyunk rosszabbnak ítélni.

5.2.2 A veszteséges minták értékelése



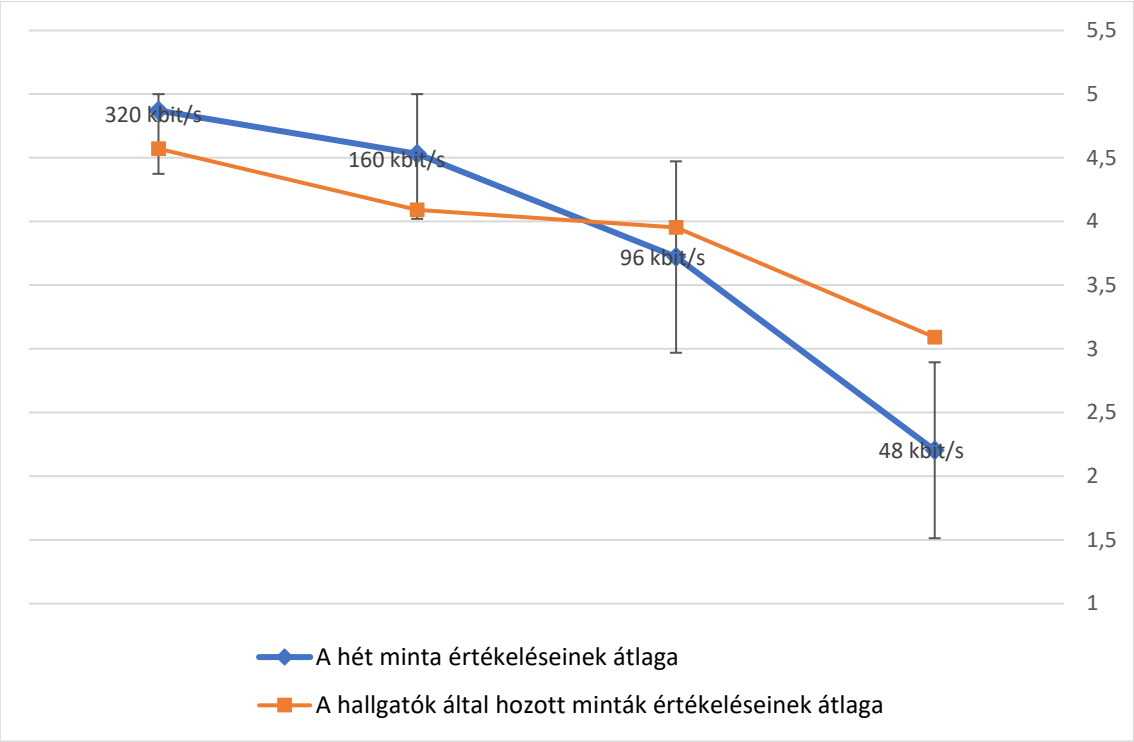
23. ábra: Szubjektív értékelés eredménye

A 23. ábrán látható, hogy a különböző hangmintákat átlagosan hogyan értékelték a hallgatók. Ezen az ábrán nem tüntettem fel az értékelések szórását, mert ennyi görbe közös ábrázolásánál ekkora méretben átláthatatlan lenne. Az ábrán látszik, hogy a legrosszabbul az 50 Hz-es fűrészjel teljesített a minőség megtartásában a bitráta csökkentése mellett. Ezt a mintát még jó minőségben sem kellemes meghallgatni, ráadásul a sok ugrás miatt érzékeny a „pre-echo” jelenségre, azaz a kvantálási zaj szétterjedésére túl nagy ablak használata esetén az ugrás előtti részekre, valószínűleg ezek járultak hozzá ehhez az eredményhez. A legjobb eredményt alacsony bitsűrűségen, az Iron érte el, de ebben nagyban szerepet játszott az, hogy ez a minta annyira hosszú, hogy mire az ember végighallgatja a veszteséges mintát, már egyáltalán nem biztos, hogy eléggé emlékszik a veszteségmentes mintára. Természetesen a szám közben is lehetett értékelni, de a hallgatók többsége a tapasztalatom szerint csak a minta vége felé értékelt, vagy miután a lejátszás befejeződött. Azt, hogy egy mintát hány másodperc után értékeltek nem tároltam el. Sajnos csak az utolsó pár teszt közben jutott eszembe, hogy ebből is érdekes következtetéseket lehetne levonni. A meghallgatásos teszt után az adatbázisból lekérdeztem a minták átlagát és szórását, amit az alábbi táblázatban foglaltam össze.

	320 kbit/s		160 kbit/s		96 kbit/s		48 kbit/s	
	Átlag	σ	Átlag	σ	Átlag	σ	Átlag	$\boldsymbol{\sigma}$
Fűrészjel	-0,19	0,85	-0,86	0,46	-2,09	0,97	-3,71	0,45
Négyszögjel	-1,14	0,63	-0,52	0,49	-1,38	0,72	-2,90	0,29
Blackbird	-0,04	0,21	-0,19	0,39	-1,00	0,61	-2,33	0,89
Castanets	-0,04	0,21	-0,47	0,49	-1,38	0,65	-3,19	0,66
Iron	-0,19	0,66	-0,38	0,57	-0,81	0,73	-1,95	0,99
Dogma	-0,23	0,68	-0,33	0,47	-0,85	0,77	-2,38	0,78
Pipe	-0,04	0,21	-0,52	0,66	-1,42	0,79	-3,09	0,75

Ezután a résztvevők sajátként hozott hangmintáit kezdtem elemezni. A minták között többféle műfaj előfordult, a legnépszerűbbek az elektronikus, az alternatív rock, és az indie folk voltak. A saját mintákra adott értékeket az adott minőségen átlagoltam, és úgy tekintettem rá, mint bármelyik másik mintára. Azért így kezeltem őket, mert így, hogy többféle műfajú és hosszúságú mintákról van szó, szerintem jó képet ad ez az ábrázolás arról, hogy valaki milyen minőségromlásra számítson átlagosan, ha úgy dönt, hogy az egész zenei gyűjteményét egy bizonyos célbitrátájú Vorbisként szeretné tárolni veszteségmentes formátumok helyett.

A hallgatók saját mintáinak átlagolása mellett a hét minta átlagos értékeinek is vettem az átlagát, hogy ezeket összehasonlítsam. Ennek az eredménye a 24. ábrán látható. A saját mintákból számolt görbénél látszik, hogy ezeknél a hosszabb és változatosabb mintáknál kevésbé érezték a különbséget a hallgatók a 160 kbit/s célbitrátájú és a 96 kbit/s célbitrátájú tömörítés között, de mind a két érték még a hét minta alapján számoltak szórásán belül van.



24. ábra: Az átlagok összehasonlítása

6 Összehasonlítás, konklúzió

6.1 PEAQ összehasonlítása a szubjektív teszttel

Ebben a fejezetben szeretném összehasonlítani a PEAQ algoritmus által kapott eredményeket az én általam lebonyolított hallgatásos tesztek eredményével.

Az első nagy különbség, hogy a PEAQ algoritmus nem olyan tesztjelek értékelésére lett kitalálva, mint például az általam használt négyszög és fűrészjel. Ezeknek az értékelésénél az algoritmus által adott eredmény nem tekinthető objektívnek. A második nagyobb – és szerintem legfontosabb – különbség, hogy amíg az embereknél rövid mintákkal kell dolgozni, hogy össze tudják érdemben hasonlítani a két jelet, addig a PEAQ modellel (elméletben) tetszőleges hosszúságú tömörített hanganyagot lehet összehasonlítani egy referenciával. Fontos különbség még, hogy ha egy hangmintán belül a hangminőség ingadozik akkor a PEAQ modell egy átlagos értéket fog adni, míg a szubjektív teszt során többször láttam, ahogy egy-egy pillanatnyi torzítás alapján (főleg 48 kbit/s-es pipe magas hangjainál) egyből rossz értékelést ad valaki, hiába volt a minta többi része megfelelő minőségű. Ez rövid mintáknál még nem okoz nagy különbséget, de hosszabbaknál biztosan.

Különbség még, hogy míg én összesen 21 emberrel tudtam meghallgattatni a mintákat, addig a PEAQ modellt nagyságrendekkel több meghallgatásos teszt eredményéből rakták össze. Ezek ellenére mind a két teszt hasonló eredményeket produkált a nem tesztjel (itt négyszögjel és fűrészjel) jellegű mintákra.

6.2 Konklúzió

Szerintem a Vorbis egy nagyon jó veszteséges audiotömörítő. Az elvégzett tesztek alapján minden fajta hangminta tömörítésénél elérhetjük, hogy megmaradjon az az érzett hangminőség amit a veszteségmentes hangminta okozott. A szakdolgozatom során végzett objektív és szubjektív tesztek alapján én úgy gondolom, hogy mindenképpen érdemesebb nagy célbitrátájú Vorbist használni valamilyen veszteségmentes formátum helyett, mert a minőségbeli romlást nem érzékeljük, viszont rengeteg tárhelyet és/vagy adatforgalmat tudunk megtakarítani vele. Ha pedig esetleg olyan célra használnánk, ahol nem annyira fontos a kiváló minőség megtartása, akkor is érdemes használni, mert a lehetőségekhez képest jó minőség érhető el vele a kódolás során használt

pszichoakusztikai modell miatt, és az eredeti méret töredékére tudjuk csökkenteni a szükséges tárhelyet, a veszteséges vektorkvantálás és az ezt követő Huffman-kódolás miatt.

Irodalomjegyzék

[1] Ogg Vorbis, Xiph.Org foundation, https://www.xiph.org/vorbis/doc/Vorbis_I_spec.html (2015. február 27.)

[2] Steganography and steganalysis: data hiding in Vorbis audio streams - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Block-diagram-of-Vorbis-encoder-43_fig14_247773842 [accessed 5 Dec, 2018]

[3] Dr. Rucz Péter: Veszteséges audiotömörítési eljárások vizsgálata, Mérési útmutató (2017)

[4] Wikipedia: Auditory Masking, https://en.wikipedia.org/wiki/Auditory_masking (revision 00:05, 13 November 2018)

[5] MathWorks: Vorbis Decoder, https://www.mathworks.com/help/audio/examples/vorbis-decoder.html, (R2018b kiadás)

[6] Thiede, Thilo & Treurniet, William & Bitto, Roland & Schmidmer, Christian & Sporer, Thomas & Beerends, John & Colomes, Catherine & Keyhl, Michael & Stoll, Gerhard & Brandenburg, Karlheinz & Feiten, Bernhard. (2000). PEAQ---The ITU Standard for Objective Measurement of Perceived Audio Quality. Journal of the Audio Engineering Society. 48. 3-29.

[7] Hydrogenandio: Vorbis, https://wiki.hydrogenaud.io/index.php?title=Vorbis (revision 21:40, 3 June 2016)

[8] Hydrogenaudio: Recommended Ogg Vorbis, https://wiki.hydrogenaud.io/index.php?title=Recommended_Ogg_Vorbis (revision 08:17, 14 October 2018)

[9] Stephen Welch, Matthew Cohen: Perceptual Coding In Python, https://github.com/stephencwelch/Perceptual-Coding-In-Python (Latest commit 2993f57 on May 14 2015)

[10] André Schild: JAVE 2, https://github.com/a-schild/jave2 (Latest commit 158d6cf on Nov 7 2018)