Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

SZEMEREY HELÉN

HANGFELVÉTELEK AUTOMATIKUS KOTTÁZÁSA

KONZULENS

Dr. Bank Balázs

BUDAPEST 2019

Tartalomjegyzék

1. Bevezetés 7

1.1. Temperált skála 8

1.2. Hangértékek és ütemek 10

1.3. Hasonló alkotások 13

1.4. Dolgozat célja 14

2. Választott eszközök 14

2.1. MATLAB 14

2.2. FL Studio 11 15

2.3. LilyBin 16

3. Tempó- és ütemdetektálás 17

3.1. Jelelőkészítés 18

3.2. Ütéskeresés 20

3.3. Tempószámítás 20

3.3.1. Hang energiájának figyelése 21

3.3.2. Ütések időbeli elhelyezkedésének vizsgálata 23

3.4. A dallam szétdarabolása 25

4. Hangmagasság detektálása 27

4.1. Első módszer — Fésűs szűrő módszer 28

4.2. Második módszer — Hangkitörléses módszer 29

4.3. Oktáv probléma 32

4.4. Harmadik módszer - Hangközablakok 32

4.5. Döntés egyértelműsége 35

4.6. Eddig ismertetett módszerek értékelése 35

4.7. Második módszer – Oktáv probléma kiküszöbölése 36

4.8. Apró módosítások 42

5. Kiértékelés 43
6. Összegfogalalás 46
7. Melléklet ismertetése 48
8. Irodalomjegyzék 49

HALLGATÓI NYILATKOZAT

Alulírott **Szemerey Helén**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet 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, 2019. 12. 12.

.....

Szemerey Helén

Összefoglaló

A dallamok lejegyzésének igénye feltehetően egyidős a dallamok létezésével. A kotta egy zenei jelrendszer, amelyet a zeneszerzők zeneművek leírására használnak. Egy zenésznek tudnia kell értelmezni a kotta jeleit, hogy a hangszerén elő tudja adni az adott zenét. A leírt kottát értelmezheti egy énekes is, aki a leírt szöveget az így leírt dallammal elénekli. A kotta használata segíti a zene változatlan megőrzését. Éppen ezért a dolgozatom célja egy olyan szoftver elkészítése volt, mely segítséget nyújt a különböző dallamok lejegyzésében.

A dolgozatom során megismerkedtem az digitális jelfeldolgozás, azon belül az audio-jelfeldolgozás lépéseivel, és egy olyan algoritmust fejlesztettem ki, mely alkalmas egy- vagy kétszólamú dallamok alapján kottaképek készítésére.

Alapvetően két paraméter ismerete szükséges feltétlenül ahhoz, hogy kottát írhassunk: a hangok időbeli elhelyezkedése és a hang magassága. Az előbbinél időtartományban, az utóbbinál frekvenciatartományban érdemes vizsgálni a különböző audio jeleket. Logikusan ebből adódik, hogy két nagyobb részre kell bontani a feladatot.

Az irodalomkutatás során már meglévő tempó- és hangmagasságfelismerő módszereket tanulmányoztam. A saját algoritmusomat ezek felhasználásával igyekeztem fejleszteni. A tervezés során több eljárást próbáltam ki és hasonlítottam össze, hogy megtaláljam a lehető legideálisabb módszert a különböző problémák megoldására.

A különböző eljárások összehasonlításához MATLAB-ot használtam. Előre rögzített hangfelvételek segítségével vizsgáltam az algoritmusok hatékonyságát, figyelve a pontosságot és a futási időt. A legjobbnak ítélt megoldást ugyanitt, MATLAB-ban implementáltam. A kottaképek grafikus ábrázolásához egy külső programot használtam, a LilyBin-t. Maga az algoritmus egy olyan kódot generál egy audio fájlból, melyet ha közvetlenül a LilyBin bemenetére írunk, a kívánt kottaképet készíti el.

Végezetül az elkészült programot teszteltem és a kapott eredmények alapján összevetettem a piacon elérhető hasonló alkalmazással. A szakdolgozat zárásaként az előállított szoftvert értékeltem és a jövőbeli továbbfejlesztési lehetőségeket is feltártam.

Abstract

The need to write melodies down is presumably as old as the existence of melodies. Sheet music is a musical signal system used by composers to describe musical compositions. A musician needs to be able to interpret the notes of a sheet music in order to play the music on his instrument. A singer who sings the lyrics with the melody so described can also interpret the recorded score. The use of sheet music helps keep the music intact. That is why the purpose of my thesis was to create a software that will help to record different melodies.

During my thesis I got acquainted with the steps of digital signal processing, including audio signal processing, and developed an algorithm suitable for creating sheet music based on one or two- part melodies.

Basically, we need to know two parameters before we can write a note: the temporal position of the notes and the pitch of the note. For the former, it is worth examining the various audio signals in the time domain and for the latter in the frequency domain. Logically, this means that the task has to be divided into two major parts.

In the course of the literature research, I studied existing tempo and pitch recognition methods. I tried to develop my own algorithm using these. During the design process I tried several methods and compared them to find the best possible solution to different problems.

MATLAB was used to compare different methods. With the help of pre-recorded music recordings, I examined the efficiency of the algorithms, observing the accuracy and the completion time. I implemented the best solution in MATLAB as well. I used an external program called LilyBin to graphically represent the sheet music . The algorithm itself generates a code from an audio file that, when copied directly to the input of LilyBin, creates the desired sheet music.

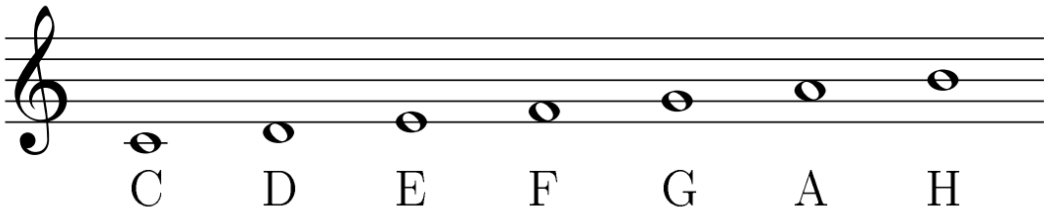
Finally, I tested the program and compared it with a similar application available on the market. As a conclusion of the thesis, I evaluated the software produced and also explored further possibilities of further development.

1. Bevezetés

A kotta zenei hangok, ritmusok és a dallamhoz tartozó egyéb kiegészítő információk lejegyzésére használatos jelrendszer. A latin kifejezés a quota szóból származik, amelynek jelentése: jegy, jegyzet, jelzés [Kotta]. A hangok lejegyzésének a célja a zenei kifejezésmódok rögzítése, átadása. Az egyértelműség érdekében a zenei alapfogalmakat Kesztler Lőrinc elmélete alapján dolgoztam fel, és ezen meghatározásokból indultam ki [Kesztler 1959].

A kottában az egymást követő hangokat, szüneteket jelölik, illetve különféle utasításokkal ezek ritmusát, játékmódját, tempóját is előírhatják. Vokális, instrumentális darabok, dallamjátszó és ritmushangszerek (pl. ütőhangszerek) játéka is rögzíthető a kottában.

A zenei hangokat az ABC nagybetűivel jelöljük a nyugati zene hangrendszerében. Alább látható a zenei hangok egy része, (a hét fő zenei hang), azok nevei, és helyük a szokásos ötvonalas rendszerben, mely öt vízszintes alapvonalból, és négy vonalközből áll. Ezeket nevezzük törzshangoknak.



1-1. ábra — Törzshangok és elhelyezkedésük az ötvonalas rendszerben

A hangokat úgynevezett „kottafejekkel” jelöljük, melyek a vonalakon és a vonalközökben helyezkednek el. A vonalak és vonalközök számozása lentről felfele történik. Az egyes szomszédos törzshangok között nem egyforma hangtávolság van, lehet félhangnyi vagy egészhangnyi távolság (értelemszerűen két félhangnyi távolság egy egészhangnyi távolságnak felel meg). A hangköz két hang magasságbeli viszonya. A nevezetes hangközöket és elnevezésüket az alábbi táblázatban foglaltam össze, ahol a hangtávolságot félhang-egységben kell érteni.

Hangköz neve	Távolság
Tiszta prím	0
Kis szekund	1
Nagy szekund	2
Kis terc	3
Nagy terc	4
Tiszta kvart	5
Bővített kvart	6
Tiszta kvint	7
Kis szext	8
Nagy szext	9
Kis szeptim	10
Nagy szeptim	11
Oktáv	12

1-1. táblázat — Nevezetes hangközök

1.1. Temperált skála

Léteznek úgynevezett származtatott zenei hangok a törzshangokon kívül, melyeket a törzshagok módosításával kapjuk módosítójelek segítségével. Három leggyakrabban használt módosító jel: ♯ (kereszt), ♭ (bé) és ♮ (feloldójel), ahol ♯ egy félhanggal feljebb, ♭ egy félhanggal lejjebb szállítja a hangot, ♮ pedig megszünteti az előző két jel hatását. Így e módosító jelek segítségével 12 hangból álló hangsorozatot kapunk, mely tartalmazza a törzshangokat, és az azokból származtatott hangokat. Minden zenei mű elemi építőkövei e 12 hang.

Megfigyelhető, hogy bizonyos hangközök sokkal kellemesebb hallgatni, mint másokat. Ha megvizsgáljuk a kellemességet, vagy a frekvenciaviszony összefüggéseit, akkor azt találjuk, hogy minél egyszerűbb a viszony, annál kellemesebb hallgatni és minél komplexebb, annál kevésbé kellemes, tehát disszonáns. A komplex hangközök azt is jelentik, hogy az adott hangok felharmonikusai közel vannak egymáshoz. Ilyen tulajdonsággal bíró két hangot együtt megszólaltatva egy periodikusan ingadozó amplitúdójú hangot hallunk. Ezt a jelenséget lebegésnek nevezzük. A lebegés frekvenciája a megszólaltatott hangok frekvenciáinak különbségével egyenlő. Ilyenkor

ezek a frekvenciák a fülben az alaphártyán is egymáshoz közel dolgozódnak fel. Ez viszont oda vezethet, hogy a két hangot nem tudjuk megfelelően diszkriminálni, így a hangok összemosódnak, disszonancia érzetét okozzák.

Léteznek úgynevezett skálák, más néven hangsorok, mely a hangok növekvő sorozatát jelentik hangmagasság szerint. A kromatikus skála egy olyan skála, amelyben tizenkét hang szerepel, és a közöttük lévő hangtávolság a félhang. Temperált skálának nevezzük az olyan kromatikus skálát, melyben minden hang egyforma félhang-távolságra van egymástól. Jellemzően a temperált skálát használjuk jelenleg a nyugati zenében. A temperált skálában az egyes félhangok $2^{\frac{1}{12}}$ távolságra vannak egymástól. Tehát ha egy adott hang alapharmonikusának frekvenciáját $2^{\frac{1}{12}}$ –vel megszorozzuk, akkor a hang felett pontosan egy félhanggal lévő hang alapharmonikusát kapjuk meg. Így ha legalább egy zenei hang alapharmonikusát ismerjük, akkor e skála segítségével meg tudjuk határozni, az összes többi zenei hangot:

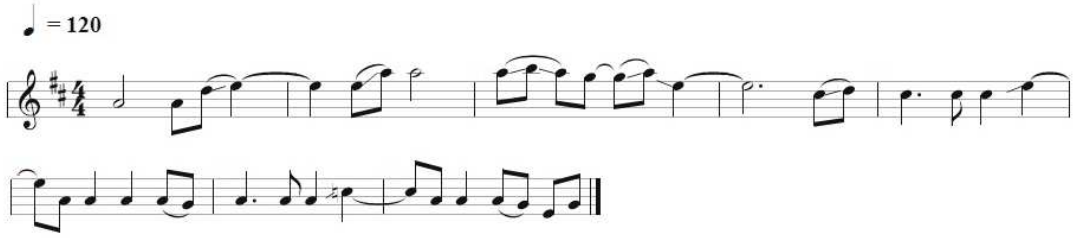
$$n \coloneqq \{1,\ \ 2, \dots, 125, 126\} \qquad \textit{és} \qquad F(n) = F_C * 2^{\frac{n-60}{12}} \qquad \textit{1. egyenlet}$$

ahol n az egyes midi kódoknak, F_c pedig a 60-as midi kódhoz tartozó hang alapfrekvenciának felel meg ($F_c = 261.63\ Hz$) [Kanizsai 2012].

A Musical Instrument Digital Interface (röviden MIDI) egy szintetizátorok és stúdióeszközök összekötésére alkalmas szabvány. 1980-ban szintetizátorgyártók szövetsége által létrehozott ajánlásként született, később vált szabvánnyá, és alkalmas a pontos zenei hangmagasság meghatározása. Maguk a midi kódok, melyeket a feladatmegoldás során fogok használni, egy táblázatban vannak összefoglalva. Ezek a kódszámok 1-től 126-ig terjedő egész számok, és mindegyik kódhoz egyértelműen hozzá van rendelve egy frekvenciaérték, és e frekvenciák mindegyikére elmondható, hogy $2^{\frac{1}{12}}$ -szerese az alatta lévő frekvenciának, tehát ez a táblázat felfogható egy temperált skálának. A számításoknál az egyvonalas C alapfrekvenciáját vettem alapul, melyhez a 60 midi kód tartozik (ezért vonom ki az aktuális midi kódból a 60-at a fentebb látható képletben). Így az *1. egyenlet* alapján valamennyi hang frekvenciája a csupán midi kódjának ismeretében egyértelműen meghatározható.

1.2. Hangértékek és ütemek

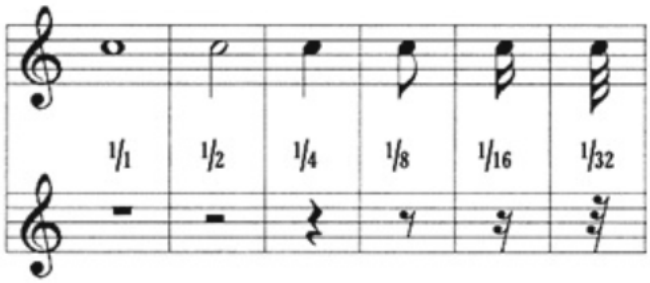
A hang magasságán kívül van még egy fontos paraméter, mely elengedhetetlen, ha egy kottát szeretnénk felírni. Ez a paraméter pedig a hang időtartama, vagy más néven a hangérték, hogy az adott hangot vagy hangegyüttest el tudjuk helyezni az ütemekben. Az ütemek a zenei lüktetést veszik figyelembe, és a zenét szakaszokra tagolják a könnyebb követhetőség kedvéért. Az alábbi ábra egy 8 ütemes dallamot mutat be, melyet ütemvonalakkal tagolunk.



1-2. ábra — Egy 8 ütemes dallam ütemvonalakkal tagolva

A 4/4-es jelölésben a nevező a lüktetés értékét mutatja (itt negyedek, de lehetnek pl. nyolcadok is: 6/8, 7/8, stb.), a számláló pedig azt, hogy egy periódusban (ütemben) hány darab érték fér el (itt 4, de lehet bármennyi: 3/4, 5/4, 7/4, stb.). Ezt a jelölést nevezik metrumnak.

Látható, hogy dallamunk különféle hangjegyértékekből épül fel, ill. szüneteket is tartalmaz. A következő ábrán (1-3.) ezek kerülnek bemutatásra. A felső sorban látható értékek nevei: egész (hossza négy negyedet ér), fél (hossza két negyed), negyed, nyolcad (egy negyedbe két nyolcad fér), tizenhatod (egy negyedbe négy tizenhatod fér), harmincketted. Az alsó sorban az értékek szünetjelei láhatók [Kanizsai 2012].



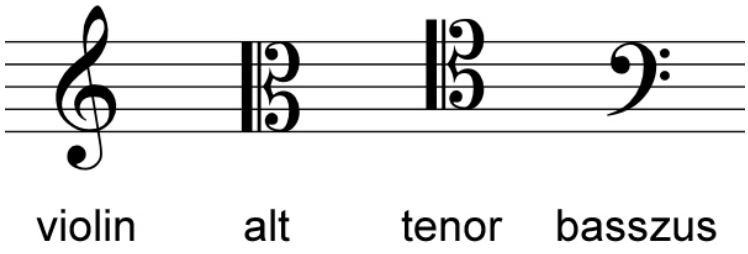
1-3. ábra — Különböző hangjegyértékek és szünetek

A negyedek hosszúsága (vagyis, hogy mennyi ideig szól egy negyed érték) mindig relatív érték, a dal tempójától függ. Egy 60-as tempónál egy negyed egy másodpercig tart; 90-es tempónál már rövidebb, így a dal is gyorsabb. (Jelölése a 1-2. ábrán is látható, ott épp 120-as tempón.)

Említsük még meg a különleges osztásokat:

- Igen gyakori a triolás osztás, ami egy negyedet két nyolcad helyett három egyenlő részre bont, miközben az egy negyednyi érték nem változik (tehát a triola gyorsabb a nyolcadnál, de lassabb a tizenhatodnál).
- A pontozott hangok értéke másfélszeresére növekszik (az első ütemben látható pontozott negyed értéke egy negyed+nyolcad értékre nő; a pontozott nyolcad értéke egy nyolcad+tizenhatodra - az utolsó nyolcadpár második hangjának értéke tizenhatodra csökkent, ezt külön jeleztük is a felső dupla vonallal).
- Az átkötött hangok értéke egybeolvad (a 1-2.ábrán a 2. ütemben látható két nyolcad közti átkötés esetén csak az első nyolcadot szólaltatjuk meg, de hossza egy nyolcaddal meghosszabbodik).

A félév során nem foglalkoztam azzal, hogy figyeljem, hogy az egyes hangok mely időpontban halkulnak el, vagyis hogy milyen hosszan tartották ki őket. Így az általam készített algoritmus szerint egy hang addig tart, amíg nem érkezik egy újabb.



1-4. ábra — Napjainkban leggyakrabban használt zenei kulcsok

Ezeket a hangértékeket el kell helyezzem egy kottában. Ahhoz persze szükséges előtte meghatároznunk a zenei kulcsot. A zenei kulcsok azok az eredetileg hangnevet jelölő betűk, ma pedig az e betűkből kialakult jelek, melyek a hangmagasság rendjét határozzák meg a zenei hangok lejegyzésére szolgáló vonalrendszerben. Régebben számos kulcs volt használatban, melyek használatát Arezzói Guidó itáliai bencés szerzetes egységesítette 1025 körül – mára azonban számuk a zenei gyakorlatban négyre csökkent (1-4. ábra), ezek közül is túlnyomórészt csak a violinkulcs és a basszuskulcs használatos [Löblin 1982].

1-5. ábra — Választott zenei kulcs és ütemmutató

A zenei kulcs mellett még az ütemmutatót is ki kell választani (1-5. ábra), ahhoz, hogy egy teljes értékű kottát felrajzolhassunk. Az általam készített algoritmusban 4/4-es ütemmutatót fogom alkalmazni, de a MATLAB kódban egy paraméter állításával könnyedén módosítható bármikor. Az 1-5. ábrán a C a 4/4-et jelöli. Bár két különböző szimbólum, a gyakorlatban ugyanazt jelentik. A C mint jelölés a múltból maradt fenn, és napjainkban mindkét jelölésformát használják [Williams 2013].

1.3. Hasonló alkotások

A számítógépek megjelenésével lehetőség nyílt a zenei művek kottaszerkesztő programok segítségével való lejegyzésére. Vannak egyszerűbb mobilapplikációk, melyek egész jó pontossággal képesek meghatározni külön-külön megszólaló hangokat, de ha egyszerre két vagy több hangot szólaltattak meg, azt már nem képesek kezelni. Ezek mobil alkalmazások egyébként sem kottázásra vannak kitalálva, magukat hangszerhangoló alkalmazásként hirdetik. Néhány példa erre: Cleartune (Android és iOS), TonalEnergy (iOS), iStroboSoft (Android és iOS).

Az automatikus kottázásra specializálódott szoftverek is elérhetőek a piacon, bár ingyenesen elérhetőt alig találni. Egy úgynevezett *AnthemScore* (melyet vezető szoftverként tartanak számon az automatikus kottázás terén) névre hallgató szoftvert demó verzióját volt szerencsém kipróbálni (sok funkciójáért sajnos fizetni kell). Kottát ír fel mp3, wav vagy más kiterjesztésű zenei fájlokból. Magáról a program működéséről, jelfeldolgozásáról nem árulnak el semmit részletesen, csupán annyit említenek, hogy az automatikus kottázást milliónyi adatmintára kiképzett neurális hálózat segítségével végzik [AnthemScore].

1.4. Dolgozat célja

A feladat kidolgozásával a célom az volt, hogy egy olyan egyszerűbb algoritmust hozzak létre, mely a bemeneti hangfelvételből (mely maximum két szólamú) egy olyan, köztes leírást hoz létre, melyből a kottázás könnyen elvégezhető. Ilyen forma például a korábban említett MIDI szabvány, melyből kottázó szoftverek segítségével pontos kottakép rajzolható ki.

A feladat elvégzéséhez szükségem lesz időtartománybeli és frekvenciatartományebeli analízisre, hiszen az időtartománybeli vizsgálat nem alkalmas az egyszerre megszólaló hangok magasságának azonosítására, míg a frekvenciatartománybeli vizsgálat az ütemdetektálást nem teszi lehetővé.

A feladat jellegéből adódóan két fő részre lehet azt osztani. Ezek a következők:

1. Ütemdetektálás, azaz a zenei hangok elkülönítése az időtartománybeli jelben.

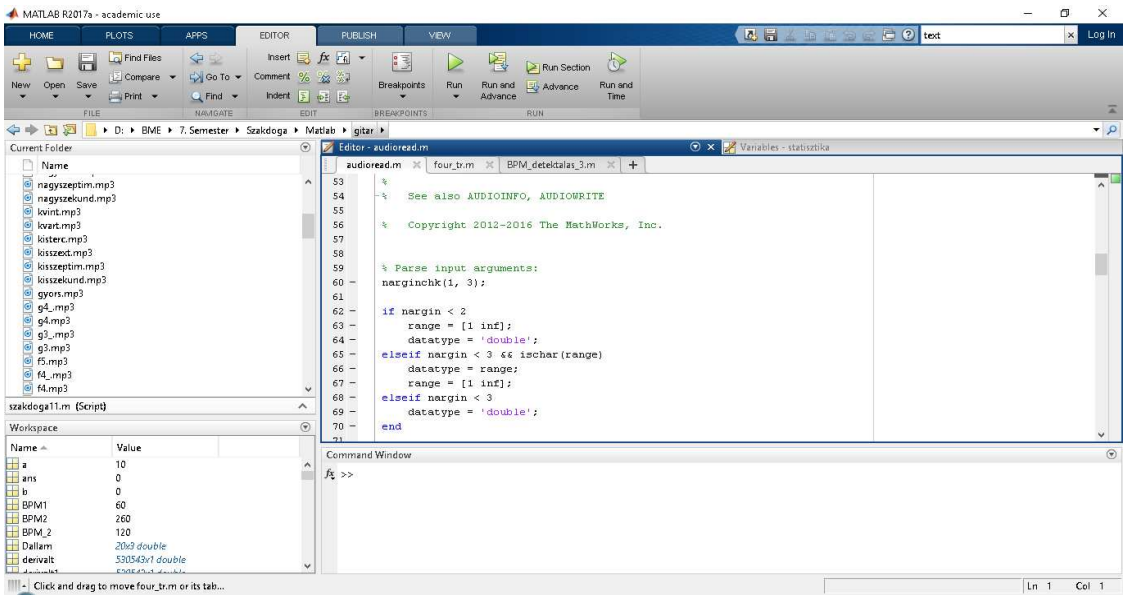
2. Hangmagasság számítása, azaz a feldarabolt mintákban az egyes megjelenő hangok szétválasztása és azonosítása a frekvenciatartományban.

A fenti felbontást követve értelemszerűen az előbbivel kell kezdeni a feladat megvalósítását.

2. Választott eszközök

2.1. MATLAB

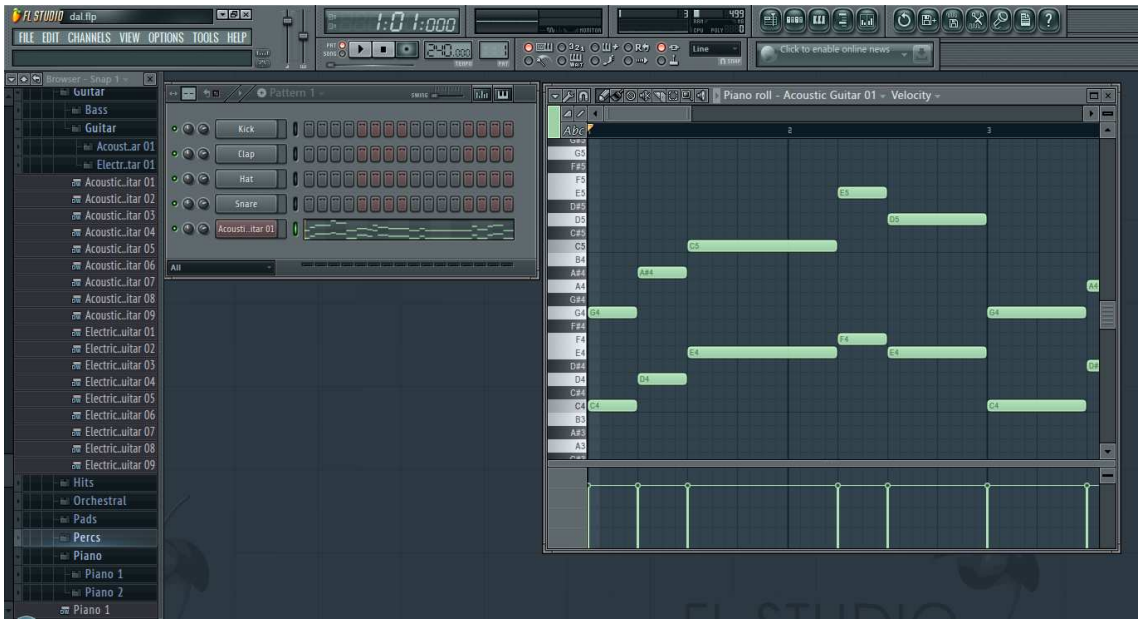
Az eljárások kifejlesztésére és tesztelésére használt eszköz a MATLAB, ami lehetőséget biztosít különböző matematikai műveletek elvégzésére és tervezési eljárások megvalósítására egy egyedi programozási nyelv használatával. A dolgozatomban látható valamennyi grafikon ezzel a programmal készült. Kezelőflülete a 2-1. ábrán látható.



2-1. ábra — MATLAB kezelőfelülete

2.2. FL Studio 11

Az FL Studio (2-2. ábra) – régebbi nevén FruityLoops – egy digitális audio munkaállomás, melyet a belga Image-Line Software fejleszt. Az FL Studio egy teljesen automatizálható munkakörnyezetet biztosít. Az alkalmazás MIDI támogatást és sok funkciót biztosít az audió szerkesztéshez, keveréshez, és felvételhez. A kész dalokat vagy klipeket Microsoft WAV-ba, MP3-ba, FLAC-ba és az OGG Vorbis formátumba tudjuk konvertálni.

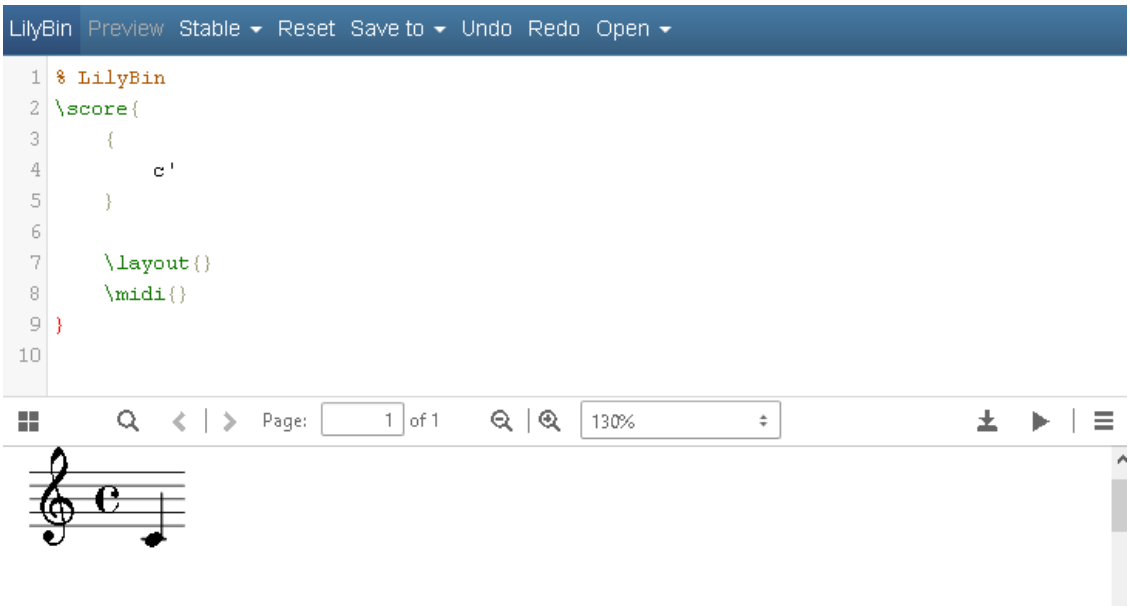


2-2. ábra — FL Studio 11 kezelőfelülete

Ezzel a programmal könnyedén generálható bármilyen zenerészlet, így a különböző referenciajeleket az FL Studio segítségével állítottam elő.

2.3. LilyBin

A LilyPond egy kottarajzolásra használható ingyenesen szoftver. Egyszerű szövegfájlokból automatikusan kottát állít elő. Maga a LilyBin egy olyan weboldal, amely kényelmes web-alapú felületet kínál a LilyPond használatához. A dolgozatomban található kottaképek nagy részét e program segítségével szerkesztettem. A 2-3. ábrán látható a LilyBin kezelőfelülete.



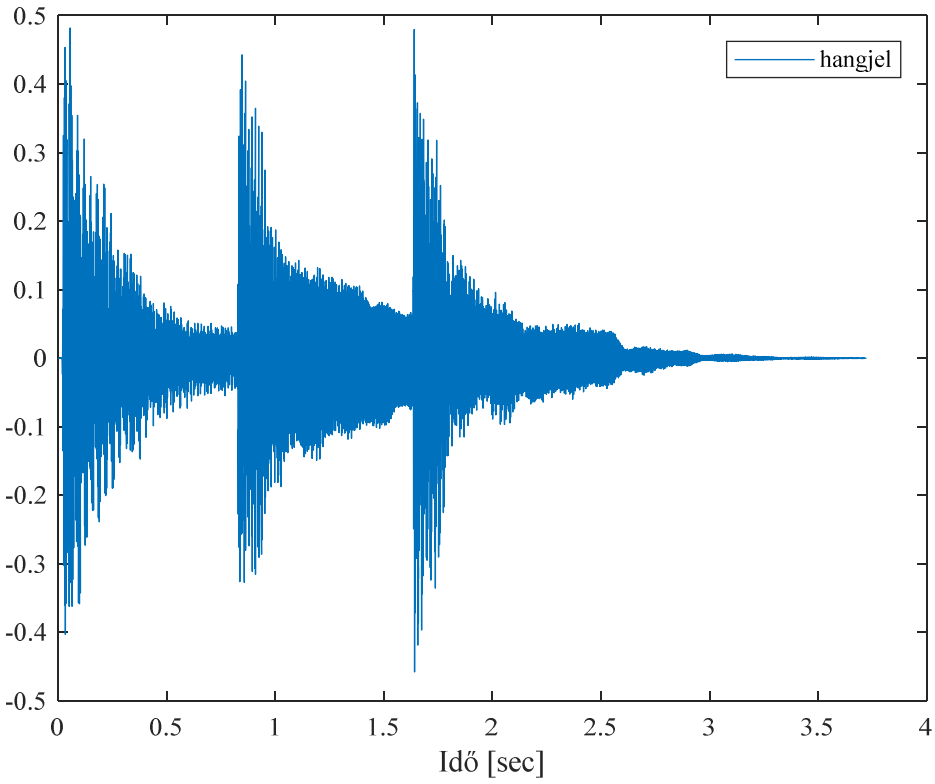
2-3. ábra— LilyBin kezelőfelülete

3. Tempó- és ütemdetektálás

Mint korábban említettem, maga az automatikus kottázás feladatát két nagyobb lépésben lehet megoldani. Az első ilyen lépés a tempó és az ütem detektálása. Erre az időtartománybeli vizsgálat az alkalmas. Az alábbi folyamatábra (3-1. ábra) bemutatja a lépéseket, melyekkel meghatározhatjuk egy többszólamú dal tempóját, melynek fontos szerepe van abban, hogy fel tudjunk rajzolni egy kottát egy hallott dallam alapján.

3-1. ábra — Tempó detektálás

Első lépésként MATLAB-ba beolvastam egy hangfelvételt, melyet 44,1 kHz-es frekvenciával mintavételeztem (3-2. ábra). E beolvasott hangfelvételt egy vektorban tároltam ezentúl, és onnan értem el, amikor szükségem volt rá, bármikor, amikor vizsgáltam, vagy számításokat akartam végezni rajta.



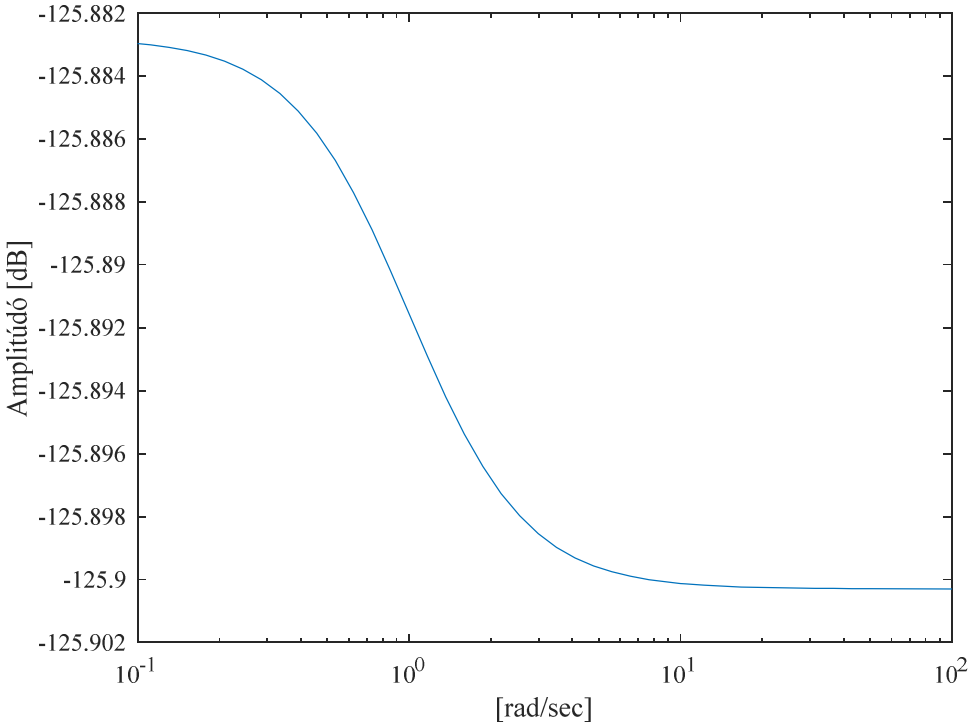
3-2. ábra — Hangfelvétel időfüggvénye

3.1. Jelelőkészítés

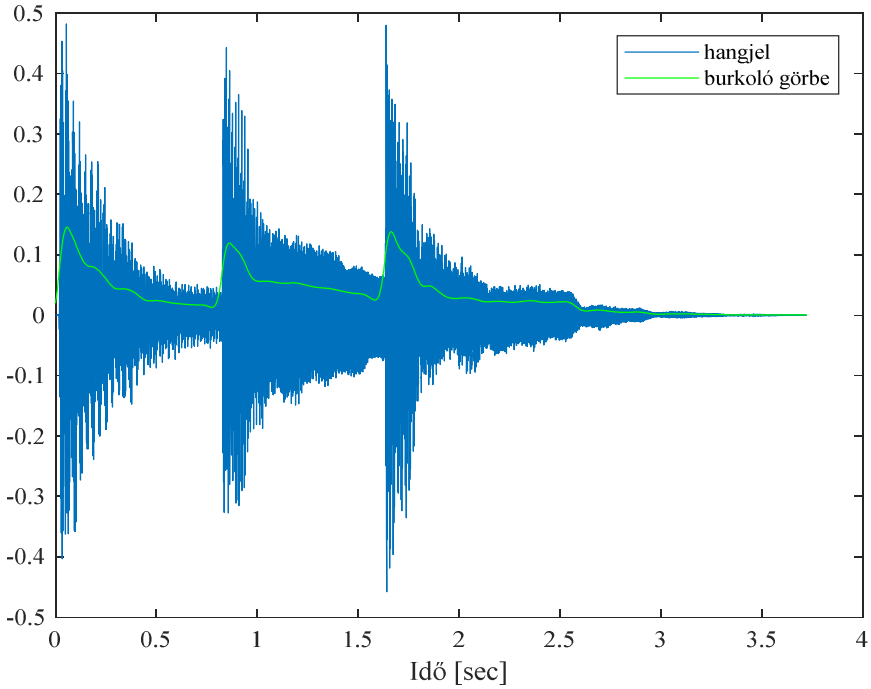
A 3-2. ábra alapján következtethetni lehet arra, hogy az adott mintában hány időben elkülöníthető hang illetve hangegyüttes szerepel. Célunk, hogy megtaláljuk, mely időpontokban ütöttek le hangokat. Ez egyszerű lokális maximumkereséssel nem valósítható meg, hiszen egy ütés sok amplitúdóváltozást tartalmaz mind a felfutás, mind a lecsengés során, azaz egy ütés több csúcsot is tartalmazhat. Az ilyen gyors változásokkal teli jelet nehéz feldolgozni, ezért a cél egy simább jel kinyerése [Balázs 2013].

Ehhez az egyik legkézenfekvőbb megoldás az aluláteresztő szűrök használata. Az aluláteresztő szűrők lényegében simítást visznek véghez a jelen, mivel kiszűrik abból a nagyfrekvenciás komponenseket. Az ideális szűrők lehetővé teszik egy adott frekvenciasáv veszteségmentes áteresztését, míg a nem kívánt frekvenciatartomány jeleit teljes mértékben elnyomják. Az ideális szűrőt nem lehet megvalósítani, ezért a cél a minél jobb közelítés megtalálása. Ezért esett a választásom a Butterworth szűrőre [Lipovszki 2012].

Széles körben használt szűrőtervezési eljárás a Butterwoth-féle eljárás. Az *N* paraméter a szűrő fokszámát határozza meg, a *Wn* pedig a törésponti frekvenciát. Minél magasabb a szűrő rendje, annál élesebb a vágás. A Butterwoth szűrőket széles körben alkalmazzák, ugyanis igen könnyen méretezhetőek. Előnyük a simaságuk és a monoton csökkenő frekvenciamenetük. A magasabb fokú szűrők megközelítik az ideális aluláteresztő szűrő frekvenciafüggvényét. A Butterwoth szűrő az IIR (Infinite Impulse Response = Végtelen impulzusválaszú) szűrők családjába tartozik. Az *a* és *b* változók a szűrőhöz tartozó együtthatók, amelyek függenek a szűrő rendjétől (*N*), a törésponti frekvenciától (*Wn*) és a szűrő típusától, ami ez esetben *lowpass*, azaz aluláteresztő. A 3-3. ábrán láthatjuk az alkalmazott — másodrendű (*N* = 2), 10 Hz-es (*Wn* = 10 Hz) — aluláteresztő szűrőt [Horváth 2017].



3-3. ábra —Alkalmazott aluláteresztő szűrő (N=3, Wn = 4.5e-4)

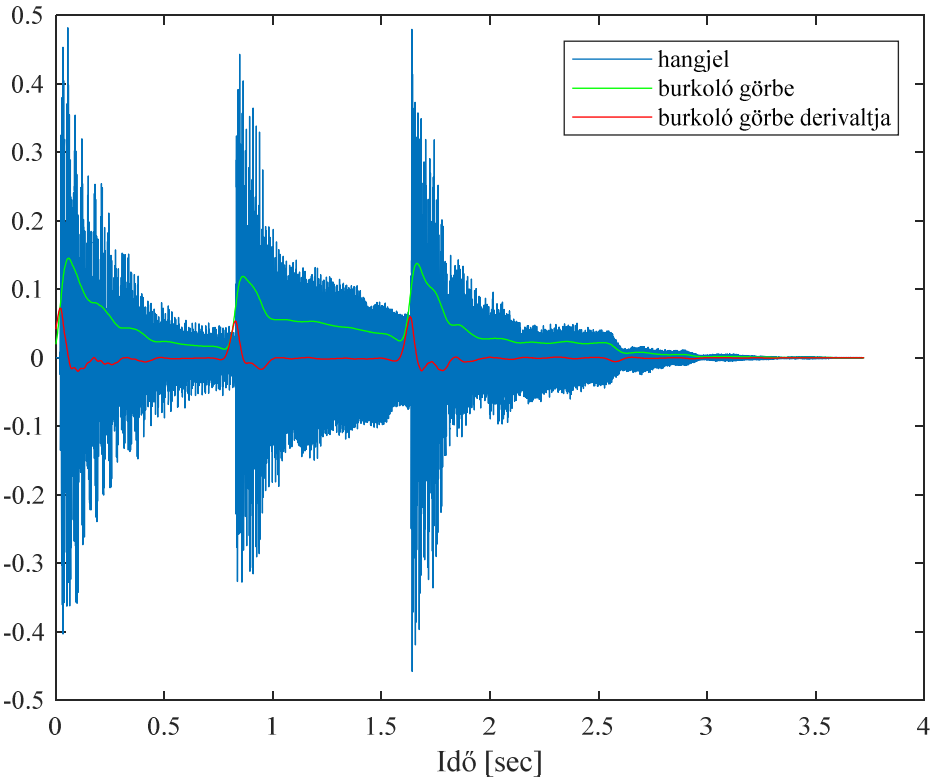


3-4. ábra — Kék- hangjel ; Zöld- hangjel burkolója

A 3-4. ábrán kékkel az eredeti jel, míg zöld színnel a szűrt hang figyelhető meg. A dallam szűréséhez az úgynevezett *filtfilt* beépített MATLAB függvényt használtam. Tulajdonsága, hogy mindkét irányba elvégzi a szűrést. Ennek eredményként nulla lesz a fázistorzítás, tehát korrigálja a szűrés során fellépő fázistolást [Horváth 2017].

3.2. Ütéskeresés

Megfigyelhetjük, hogy azokat a pontokat keressük (ütéseket), ahol a burkoló a legmeredekebb. E pontok meghatározásának legegyszerűbb módja, ha deriváljuk a burkolót. Tudjuk, hogy egy függvény első deriváltja ott a legmagasabb, ahol az adott



3-5. ábra — Kék színnel-eredeti jel; Zöld színnel-az eredeti jel burkolója; Piros színnel a burkoló első deriváltja

függvény a legmeredekebb, így a 3-5. ábrán látható is, hogy a burkoló első deriváltjának csúcsai fogják adni az ütések helyét**.** Hangegyüttesek leütésének helye ezzel körülbelül századmásodperces pontossággal meghatározható.

3.3. Tempószámítás

Az ütések helyének meghatározása után a tempó értékét szükséges kiszámolni. A kottában az egymást követő hangokat, szüneteket jelölik, illetve különféle utasításokkal ezek ritmusát, játékmódját, tempóját is előírhatják. A tempó kiszámítása egy fontos lépés, ha kottát akarunk készíteni, hiszen ez alapján határozható meg, hogy időben milyen gyorsan követik egymást a hangok, illetve hogy a különböző hangértékek milyen hosszan vannak kitartva.

A hangok hossza a zenében általában egyszerű arányokkal megadható, ahogy azt már láthattuk az 1.2 fejezetben. A legismertebb hangértékek a következők: egészhang, félhang, negyedhang, nyolcadhang. Ezt a sort tovább is lehetne folytatni. Egy egész két félből áll, egy fél két negyedből, egy egész négy negyedből, és így tovább. A hangok értékei azonban csak az egymáshoz való viszonyukat fejezik ki. Hogy pl. milyen hosszú 1 negyed hang, ahhoz már szükséges valamiféle tempójelzés megadása is. Egy zenemű tempóját alaplüktetése határozza meg. Ezt a lüktetést valamilyen egyenletesen érkező fizikai esemény (pl. hang) miatt érezzük. Ha a ritmus egyes hangjai megerősítik a lüktetést, akkor a tempó világosan érezhető. A tempót a hallható vagy csak a tudatban jelenlévő lüktetés elemei közötti időtartamból lehet kiszámolni. A tempót meg lehet adni a lüktetést kiváltó hangok között eltelt idővel (sec vagy msec) vagy a percenkénti ütésszámmal (BPM – beat per minute) [Szigetvári 2014]. Az utóbbival írtam le későbbiekben a tempót.

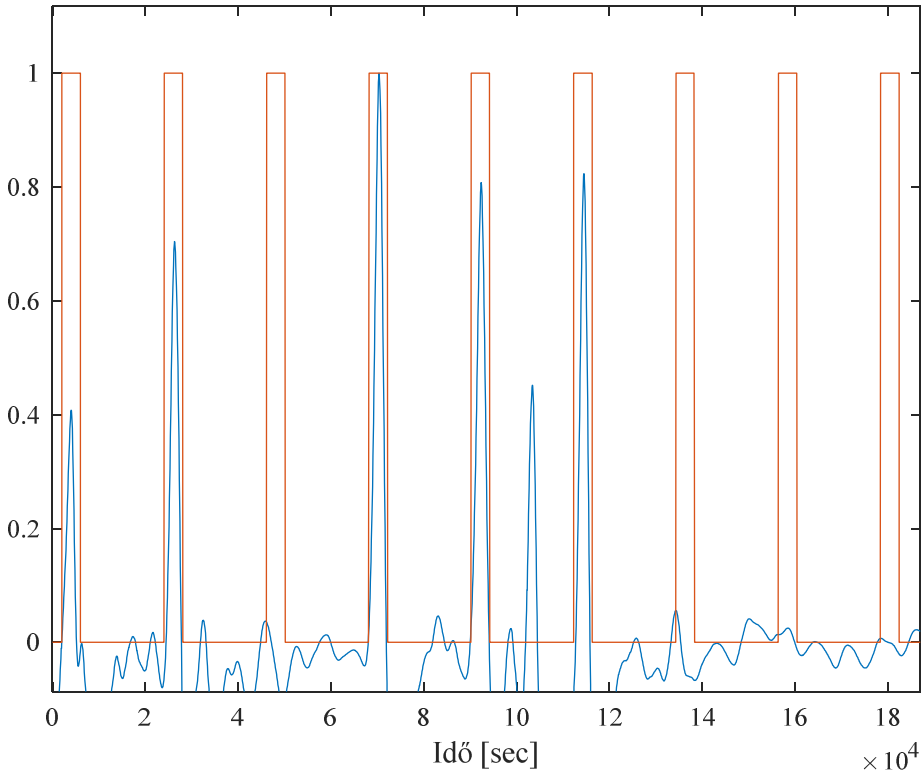
A kiindulási állapotban rendelkezésre állnak a jelben lévő ütések és azok helyei századmásodperc pontossággal. Érdemes egy pár másodpercnyi mintát kivágni a dallamból, és nem az egészet vizsgálni, hogy elkerüljük a sok és felesleges számítás. Elég kiragadni néhány másodpercet a hangfelvétel közepéből. Általában kb 4 másodpercet vágtam ki az egyes dalokból, így a mintákban legalább három lüktetés volt a leglassabb tempó esetében is, ami jelen esetben 60 BPM (tehát átlagosan 1 másodpercenként érkezik új hang).

A BPM értékét a következők szerint határoztam meg: minden BPM értékhez létrehoztam egy szűrőt, mely segítségével vizsgáltam a hang energiáját. Majd továbbra is ennek az elvnek alapján próbáltam javítani, finomítani az eredményeken, és gyorsítani a futási időn.

3.3.1. Hang energiájának figyelése

Az algoritmus elején meghatároztam két konstanst, BPM alsó és felső határát, így az átlagos másodpercenkénti ütések számát két meghatározott érték között keressük. Jelen esetben ez a két érték 60 és 240 BPM, hiszen a különböző dalok tempója jellemzően e két érték közé esik.

E két adott BPM érték között egyesével végighaladva, mindegyik tempóértékhez létrehoztam egy fésűs szűrőt, melynek segítségével a szűrt jel első deriváltját vizsgáltam (3-6. ábra) [Horváth 2017].

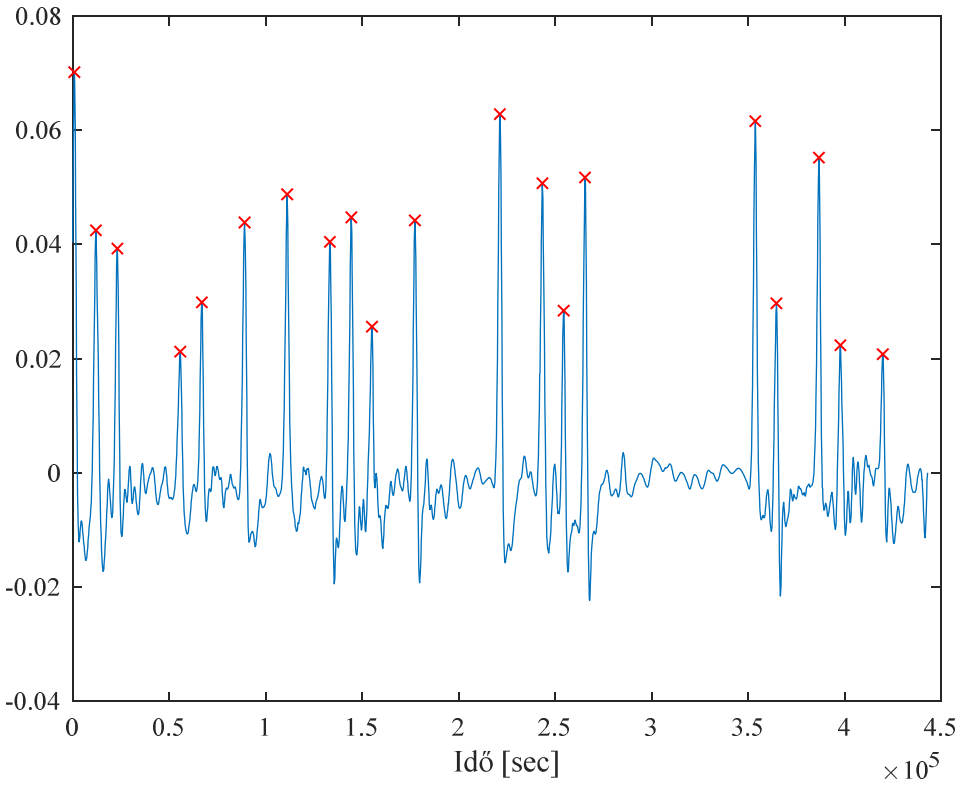


3-6. *ábra — Kék: szűrt jel első deriváltja, Piros: fésűs szűrő*

Maga a BPM csak azt mondja meg, hogy milyen távol legyenek egymáshoz képest az ablakok a szűrőben, de nem tudjuk megmondani, hogy hol helyezkedjen el a szűrt jelhez képest. Emiatt a szűrőt (minden BPM érték esetében) tologatni kell, és figyelni, hogy hol adja a legnagyobb energiát, ezzel garantálva, hogy legalább egy ütés helyére illeszkedjenek egy ablak a szűrőből. Ez a fajta BPM meghatározás elég lassúnak mondható, körülbelül 15 másodpercre van szüksége az algoritmusnak. Ezért megpróbáltam alulmintavételezni a szűrt jelet, de ha csak minden 100. mintát vizsgáltam, akkor is 8-10 másodperc volt a futási ideje a függvénynek, amely elvégzi a BPM detektálást.

3.3.2. Ütések időbeli elhelyezkedésének vizsgálata

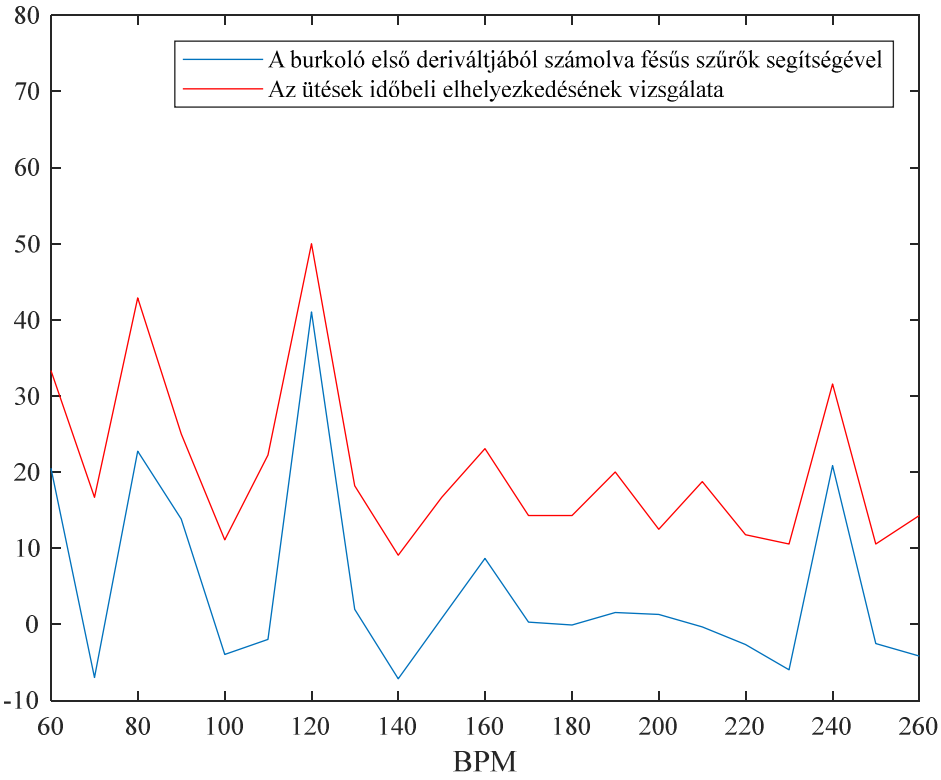
Korábban kiszámoltam az első derivált alapján a dallam összes hangjának és hangegyüttesének helyét (emellett mindnek eltároltam mellé a maximumértékét), ehhez pedig tökéletesen lehet viszonyítani a fésűs szűrő elhelyezését. A kivágott mintában a legsúlyosabb ütés helyére rá lehet illeszteni az első ablakot, ebből kiindulva pedig a többi helyét az aktuális BPM értéknek megfelelően határoztam meg. Ezzel, hogy korábbi számításokat felhasználtam, fél másodpercre sikerült redukálnom a BPM detektáló függvény futási idejét.



3-7. ábra — Ütések jelölése a szűrt jel deriváltjában

Az eddigiekhez hasonlóan ennél a módszernél is használtam ablakokat, de nem számoltam ki vele a jelnek az energiáját. Vehetjük úgy, hogy minden egyes ablak egy-egy tartományt jelöl ki, és szimplán csak annyit vizsgáltam, hogy hány ilyen időtartományban található 1-1 ütés. Természetesen minden BPM értékhez eltároltam az ütések számát, melyek a kijelölt tartományba estek.

A 3-7. ábrán kékkel a hangfelvétel közepéből kivágott mintának az első deriváltja látható, pirossal pedig maximumkereséssel megtalált ütéseket jelöltem. Ez a módszer hasonlóan jó eredményt hozott, mint az energiaértékek számítása (3-8. ábra), de jelentősen kisebb volt a futási ideje.



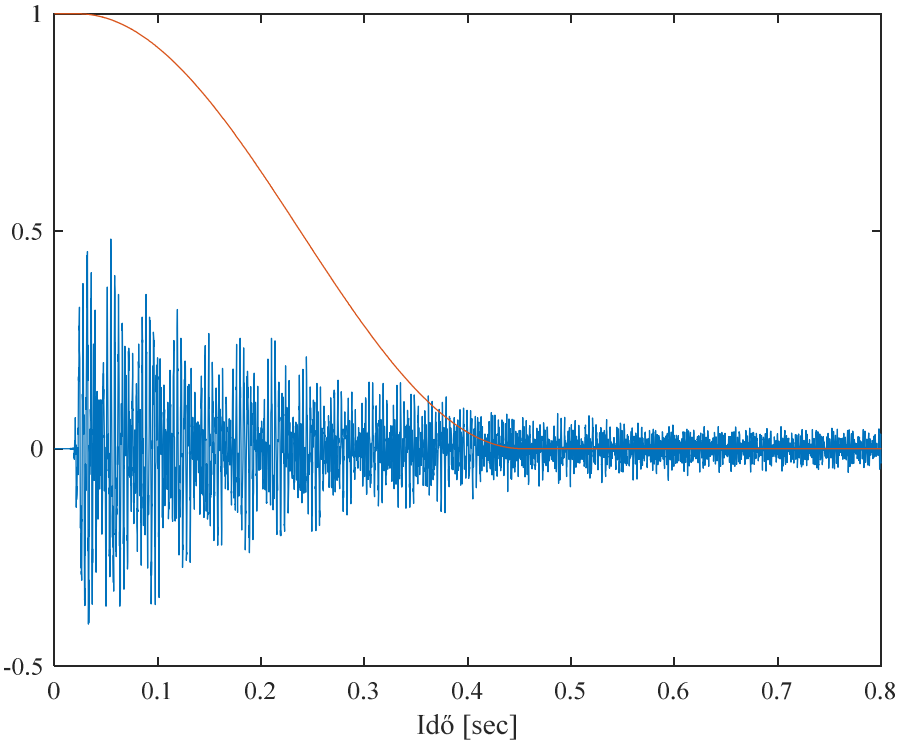
3-8. ábra — A két tárgyalt BPM számítási módszer eredménye ugyanazon a dallamon

Most, hogy már ismerjük a tempót, könnyedén kiszámíthatók a hangértékek. Tételezzük fel, hogy a tempó 120 BPM, tehát percenként 120 lüktetésünk van, azaz fél másodpercenként van egy lüktetés. Ez fogja adni a negyedhang hosszát, az összes többi hangérték pedig ehhez viszonyítva megadható.

Meghatároztam egy tetszőleges dallam ütéseinek helyét, tempóját, egyes hangok vagy hangegyüttesek hosszát, és azok időbeli elhelyezkedését, majd azokat átskáláztam úgy, hogy a megszólaltatás időpontja ne másodpercben legyen megadva, hanem ütemekben kifejezve. Ezek után az a teendőnk, hogy minden egyes ütésnél meghatározzuk a hangmagasságot. Tudjuk, hogy egynél több hang is szólhat egyszerre, így az időtartománybeli vizsgálat nem alkalmas a hangmagasság detektálására, tehát innentől kezdve a spektrumát kell vizsgálnom, amit Fourier transzformációval kapok meg. Ehhez azonban előtte szét kell darabolni időben a dallamot annyi részre, ahány ütés történt.

3.4. A dallam szétdarabolása

A beolvasott dallam szétdarabolása még időtartományban történék. Az ütések helyeit ismerjük. Tehát ha két ilyen detektált hely közötti részt vágunk ki, akkor az már rossz megoldásra nem vezethet, az algoritmus későbbi részeiben nem okoz hibát. Viszont figyelembe kell venni ebben az esetben a számításigényt. Tételezzünk egy 60 BPM- es tempót. Ekkor egy negyedhang 1 másodperc hosszú, amiből következik, hogy egy egészhang 4 másodpercig tart. Ez nagyon hosszú idő, ha így mintavételezünk, a futási idő szükségtelenül nagy lesz. Egy hosszabban kitartott hang vagy hangegyüttes esetén túl



3-9. ábra — Egy megszólaltatott hangegyüttes és a hozzá tartozó Hann ablak

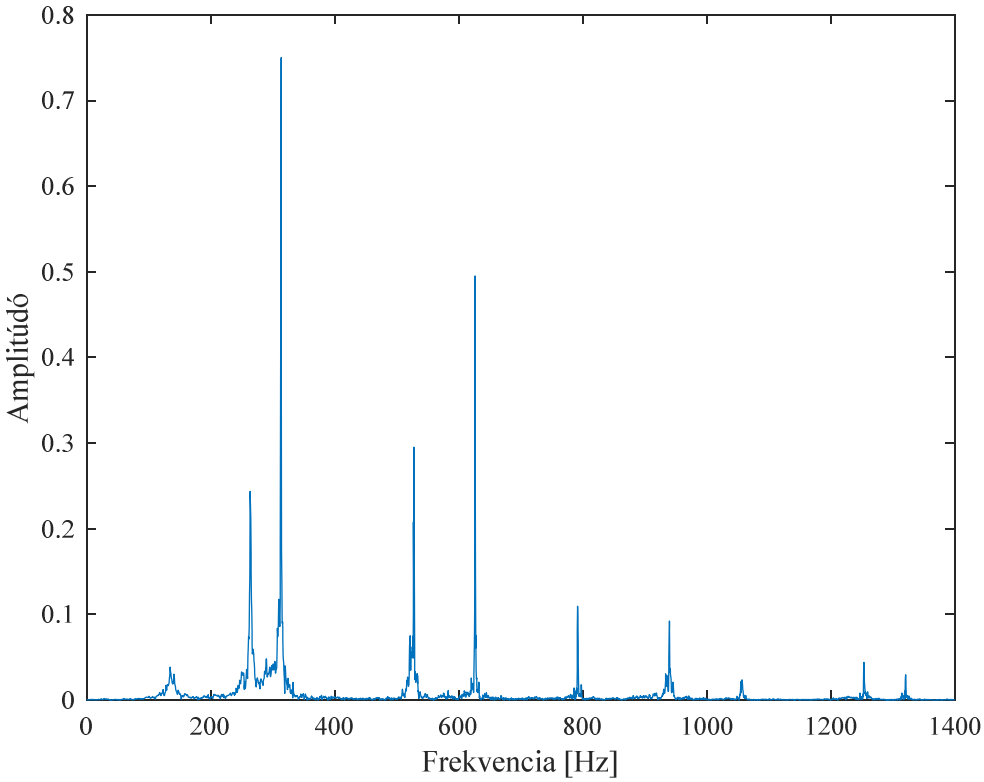
nagy lenne így a program számításigénye. Keresni kell valami szűkebb időtartományt, amellyel a hangot vizsgáljuk, és még nem ront túl sokat a hang frekvenciaképén.

Több teszt futtatása után, ahol különböző méretű ablakokkal vágtam ki egy-egy szeletet a dalból, arra az eredményre jutottam, hogy körülbelül az ütés és az ütéstől számított 0,3 másodperc között idő frekvenciatartománybeli vizsgálata még elég pontos eredményhez vezetett, nem rontott a későbbi számításokon. Bár ettől rövidebb minta vizsgálata már kevésbé lesz pontos, egy gyorsabb tempójú dallamnál, ahol két hang között kevesebb, mint 0,3 másodperc telik el, elengedhetetlen, hogy szűkebb tartományt vizsgáljunk. A

szükséges rész kivágásához a hangfelvételt egy változtatható hosszúságú fél *Hann ablakkal* (3-9. ábra) szoroztam meg, hogy az ütés körüli részt hangsúlyozzam ki, hiszen az hordozza a legértékesebb információkat. Ezek után e „kivágott” hangokból vett spektrumok alapján a hangegyüttesek hangmagasságát kell meghatározni, melyet a következő fejezetekben fogok tárgyalni.

4. Hangmagasság detektálása

Az előző fejezetben feldarabolt mintákat használtam fel a hangfelvétel kottázásához. Az egyes mintarészekben ismeretlen számú egyszerre szóló hang lehet jelen. Ezek elkülönítése időtartománybeli számításokkal nem lehetséges, így a minták spektrumát szükséges vizsgálnom, melyet MATLAB segítségével képeztem *fft* függvénnyel.



4-1. ábra — Hangminta spektruma (c4 – dis4 hangegyüttes)

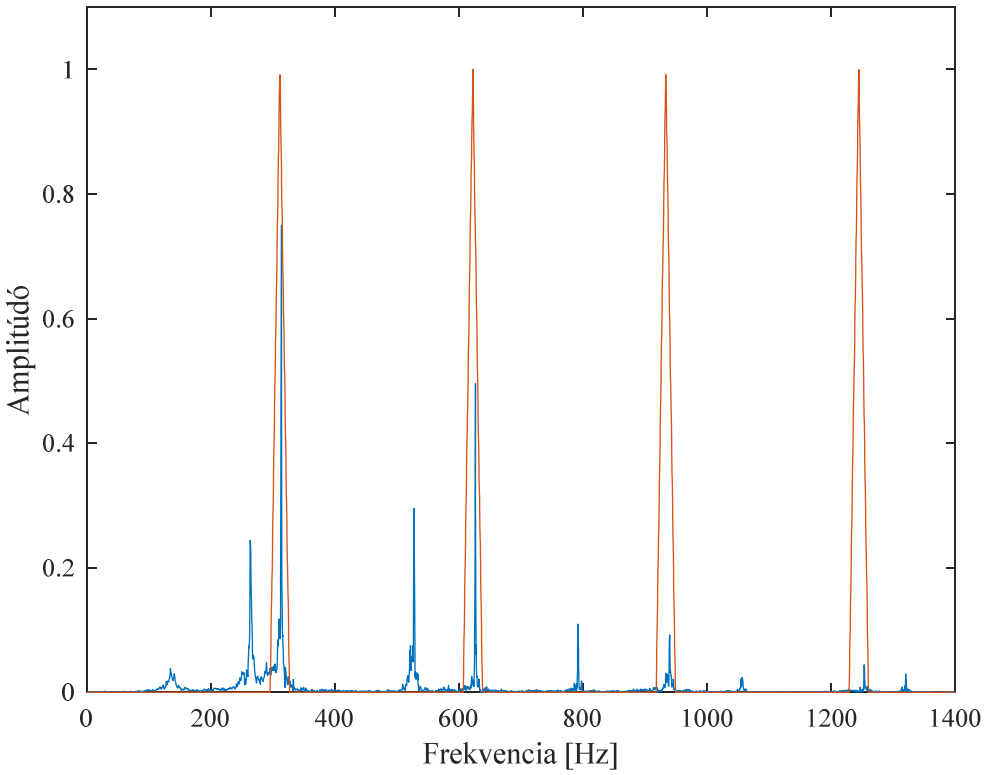
A 4-1. ábrán egy hangegyüttes frekvenciatartománybeli képét láthatjuk. Általában decibelben szokás ábrázolni a spektrumot. Jelen esetben azért nem így teszek, mert így jobban kiemelkednek az erősebb komponensek, ezért a későbbiekben sem decibelben számolok. Tudjuk, hogy a harmonikus hangok egy tulajdonsága, hogy a felharmonikusaik egy alapharmonikus egész számú többszörösének megfelelő frekvenciákon jelennek meg. Mivel az 4-1. ábrán megfigyelhetjük, hogy a csúcsok egy-egy csoportja egyenletesen oszlik el a frekvencia tengely mentén, ezért arra tudunk következtetni, hogy harmonikus hangok vannak jelen e mintában.

Egy harmonikus nem határoz meg egy hangot, ezért harmonikusok csoportját keressük. Erre célszerű egyes zenei hangokhoz szerkesztett fésűs szűrőket alkalmazni, mellyel a frekvenciatartományban lehetséges a minta ablakozása az egyes hangoknak megfelelően

[Zheng 2007]. A fésűs szűrőkhöz először szükséges meghatározni a zenei hangok alapharmónikusait. Ehhez használtam a midi táblázatot, melyben minden zenei hanghoz egy sorszám van párosítva. Elegendő így egy hang alapharmónikusának a frekvenciáját ismernünk, a többi alapharmónikus már egy egyszerű képlettel meghatározható, hiszen tudjuk, hogy az egyes félhangok $2^{\frac{1}{12}}$ távolságra vannak egymástól.

4.1. Első módszer — Fésűs szűrő módszer

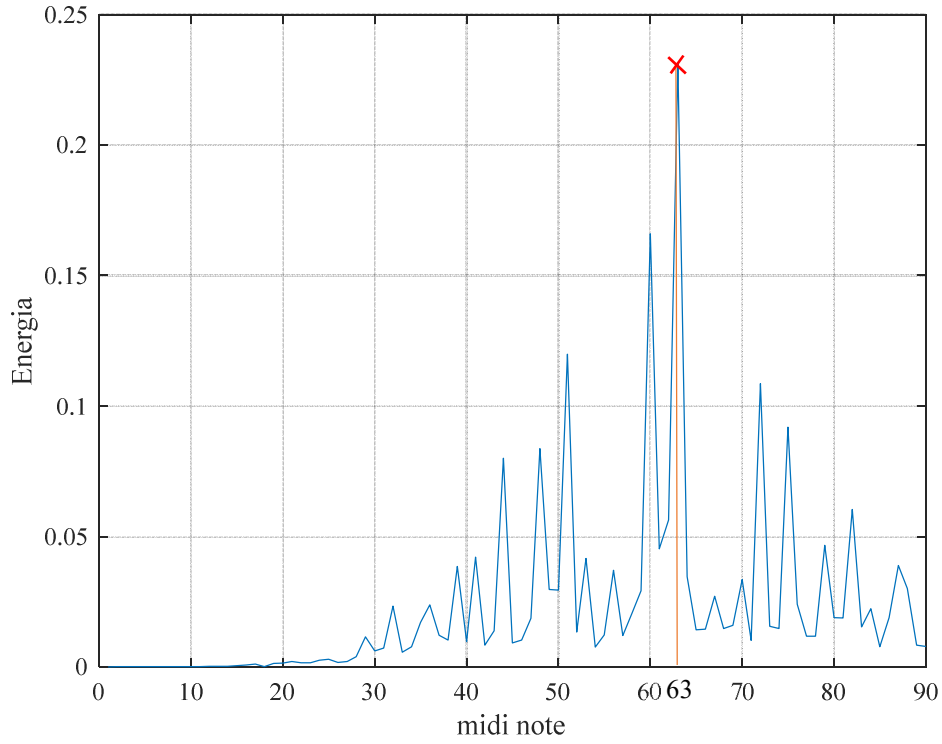
Ezután a zenei hangokon végighaladva létrehozom a hozzájuk tartozó fésűs szűrőket, ami harmonikusoknak megfelelő, az ideális frekvencia, azaz az egyes zenei hangok alapharmónikusai alatt és felett 5%-kal kezdődő, illetve végződő háromszög alakú ablak, ahogy az az 4-2. ábrán látható.



4-2. ábra — Fésűs szűrő alkalmazása (c4 – dis4 hangegyüttes)

Minden egyes zenei hanghoz generáltam egy hasonló fésűs szűrőt, így ezt követően az egyes hangokhoz eltároltam a spektrum és szűrő szorzatát, az így kapott vektor elemeit összeadtam, és ezt az összegértéket az adott hangmagassághoz rendeltem. Miután végighaladtam ennek megfelelően az összes zenei hangon, kirajzolódott egy függvény

(4-3. ábra). MATLAB segítségével meghatároztam e függvény maximumértékét, majd az ehhez tartozó hangot választottam 1. számú megszólaltatott hangnak.



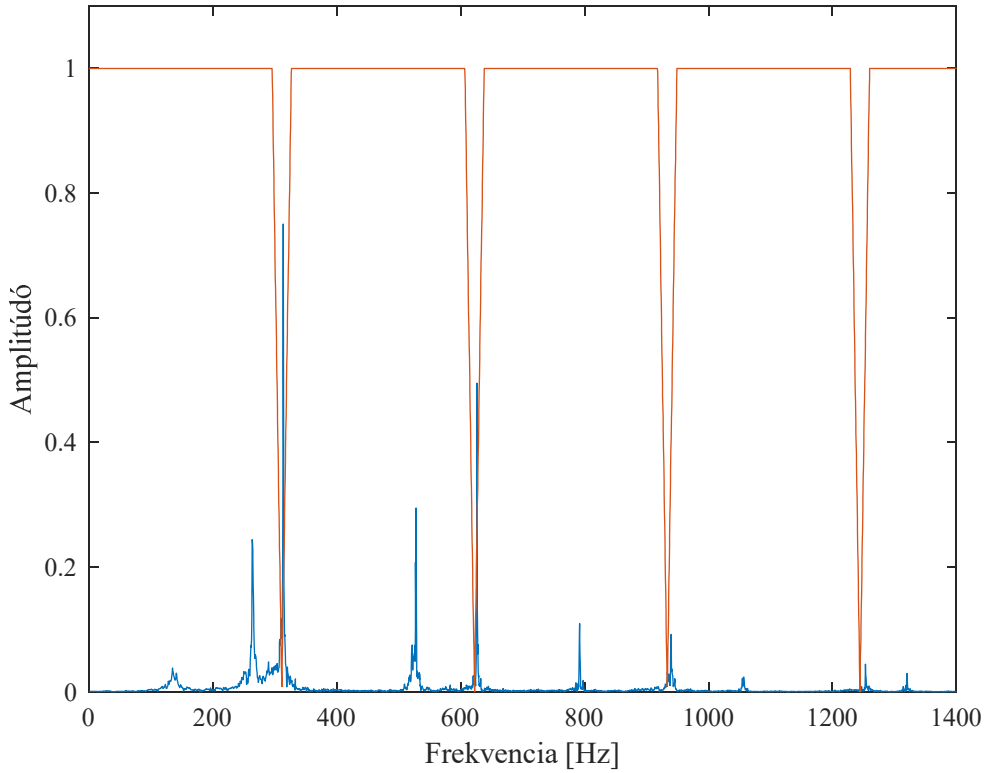
4-3. ábra — Legsúlyosabb hang keresése (c4 – dis4 hangegyüttes)

Ez a klasszikus gitár hangtartományában, amennyiben tudjuk, hogy csak egy hangot szólaltattak meg, mindig jó eredményt adott. Ha tudjuk, hogy egyszerre két hangot ütöttek le, a második legnagyobb összeghez tartozó hangot tüntettem ki a 2. számú megszólaltatott hangnak fésűs szűrők módszerét alkalmazva. Ezt az úgynevezett 2. számú hangot azonban mindössze 60% pontossággal tudta helyesen meghatározni az algoritmus. Általában ilyenkor a keresett hang egy oktávját, vagy éppen az első számú hang oktávját találta meg. Ennek az lehet az oka, hogy az 1. számú felismert hang sokkal hangosabban szólalt meg, mint a másik, ezért az egyes hang oktávja is nagyobb összeget adott, mint a keresett hang.

4.2. Második módszer — Hangkitörléses módszer

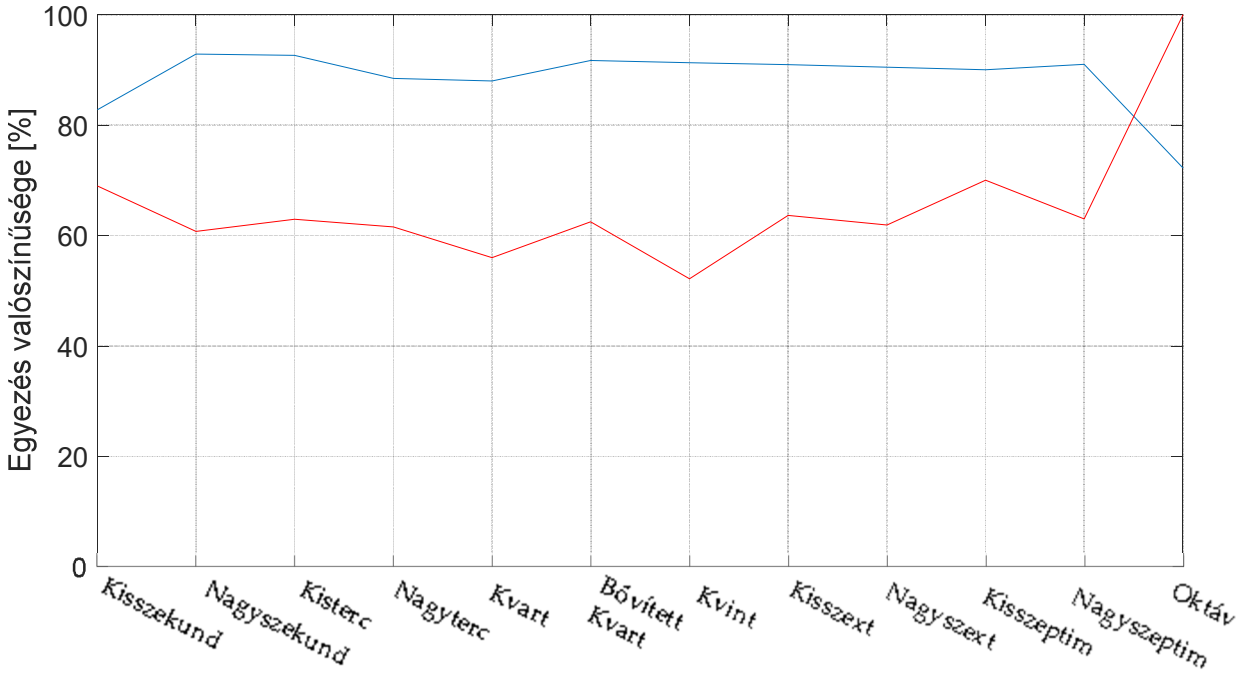
Mivel Fésűs szűrők módszere minden esetben helyesen találta meg az egyik hangot, amelyet az 1. számú felismert hangnak neveztünk, így eddig a részig változatlanul hagytam az algoritmust. A második hang keresésekor az első hang harmónikusai megzavarták a döntést, hiszen tartalmazza saját oktávjának harmonikusait. Tehát így a

célom, hogy az 1. számú hangot figyelmen kívül hagyjam, amikor a másodikat keresem. Erre a legkézenfekvőbb megoldás, ha egyszerűen kivonom a spektrumból az 1. számú hang harmonikusait. Jelen esetben ezt úgy oldottam meg, hogy fordított háromszögablakokkal szoroztam meg a spektrum megfelelő részeit, ahogy az a 4-4. ábrán is látható.



4-4. ábra — Az első számú felismert hang spektrumból való kivonása

Tesztelésképpen a klasszikus gitár hangtartományában (E2-C6) - kisszekundtól az oktávig- képeztem az összes hangközt, majd mindkét módszer alkalmazása után kiértékeltem az eredményt. A fésűs szűrők módszere futási időben sokkal kedvezőbb volt, hiszen csak egyszer kellett végighaladni a fésűs szűrők segítségével az összes hangon, viszont kevésbé volt pontos. Átlagosan 60%-os pontossággal találta el a második hangot (4-5. ábra).



4-5. ábra— Az első (piros) és a második módszer (kék) pontossága százalékban kifejezve

A hangkitörléses módszer habár nagyobb számításigényű, de láthatóan jobb eredményre vezetett, kb 90% pontossággal találta el a hangközöket, amennyiben tudtuk, hogy két hangot ütöttek le egyszerre.

Megfigyelhető, hogy a második módszer átlagosan jó eredményt ad vissza, de speciálisan az oktávnál ez kevésbé igaz. Ennek oka, hogy az első számú hang törlésénél adatvesztés lép fel, hiszen olyan harmonikusokat törlök, amelyek a másik hangnak is részét képezik, vagy esetleg teljesen el is tűntetem ezzel a második számú hangot.

4.3. Oktáv probléma

A 4.1 fejezetben bemutatott fésűs szűrők módszerének legfőbb problémája, hogy a keresett hangköz helyett valamely megszólaltatott hangot és annak oktávját találja meg (és ez a probléma különösen gyakori, ha az egyik hangot hangosabban szólaltattak meg, mint a másikat), mert a hangok oktávainak harmonikusai egymásnak többszörösei vagy egybeesnek, így a mélyebb hang távolabbi harmonikusainak is tűnhetnek. E tulajdonsága miatt maga a szakirodalom is kiemelten kezeli az oktávot [Antony 2009].

A 4-5. ábran láthatjuk, hogy az első módszer 100%-os pontossággal találta meg az oktávot, vagyis minden alkalommal, amikor oktávot szólaltattunk meg, valóban oktávot érzékelt. Viszont sok más hangközt oktávnak hitt az algoritmus, ami a fent leírt tulajdonságból következő probléma.

A második módszer, miszerint az első számú megtalált hangot kitöröljük, majd újra keressük a legerősebb energiájú hangot, orvosolta azt a problémát, hogy ne higgye oktávnak az olyan hangközöket, amelyek nem azok. Hiszen a hang harmonikusai kitörlésével, annak oktávjának harmónikusait is eltüntettük. Ez pedig egy olyan problémát szült, amely az első módszerben működött jól. Az oktávok detektálást drasztikusan lerontotta. Azzal, hogy az elsőszámú megtalált hang harmónikusait kitöröltük a spektrumból, azzal annak a hangnak az oktávjának a harmónikusainak nagyrészét is kitöröltük, így szinte már csak zaj maradt utána.

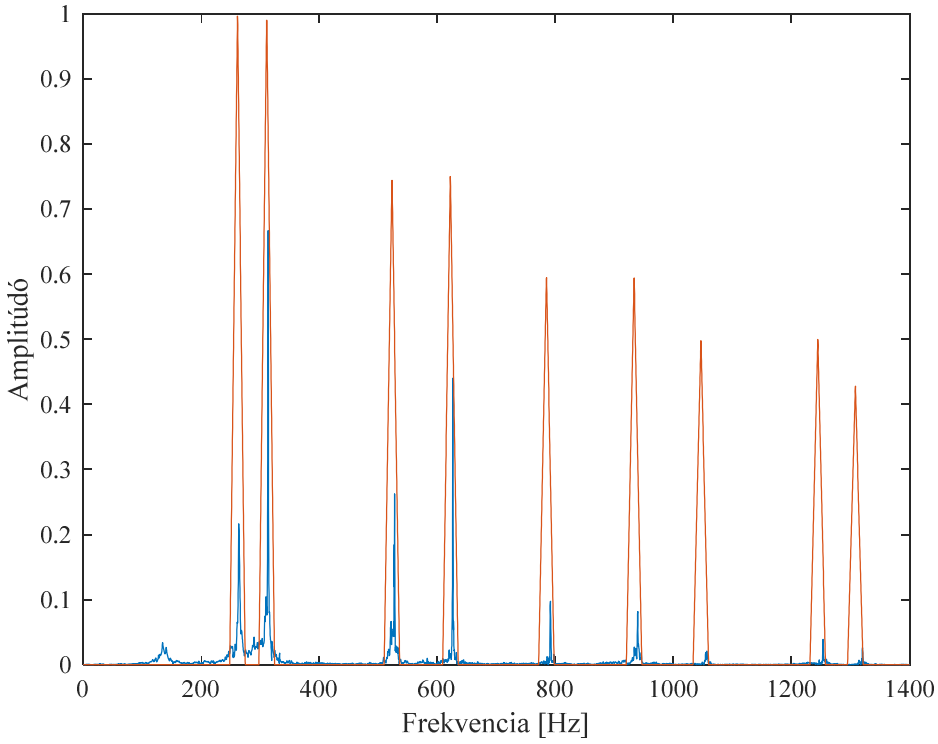
Így van két algoritmusunk eddig, melyből az egyik az oktávot képes pontosan megtalálni, de a többi hangközt nem, a másik, mely az oktávot ritkán tudja felismerni, míg az összes többi hangközt igen. Ezért szükséges egy olyan megoldást találni, mely minden hangközt képes megközelítőleg azonos pontossággal megtalálni.

4.4. Harmadik módszer - Hangközablakok

Eddigiekben láthattuk, hogy a hangok egyesével való kereséséből, elemzéséből nem tudtuk elég pontosan meghatározni a hangmagasságot egyszerre két hang megszólaltatásakor. Vagy a tiszta hangközök spektrumát rontottuk el, vagy az összes többiét. Így szükségét éreztem, hogy egy új algoritmust dolgozzak ki, egy ún. harmadik módszert, melynél a hangközöket nem úgy próbálom meghatározni, hogy az őket alkotó hangokat keresem meg egyesével, hanem közvetlenül adott hangközöket keresek. Magyarul a fésűs szűrőket már nem egy-egy hangra illesztem, hanem egyszerre kettőre.

Tehát, ha keresek egy kis tercet (3 félhang távolság), akkor generálok az alsó hangra, aztán a felette 3 félhanggal lévő hangra is egy fésűs szűrőt, majd ezt a két szűrőt összeadva, azután a spektrummal megszorozva vizsgálom azt (4-6. ábra).

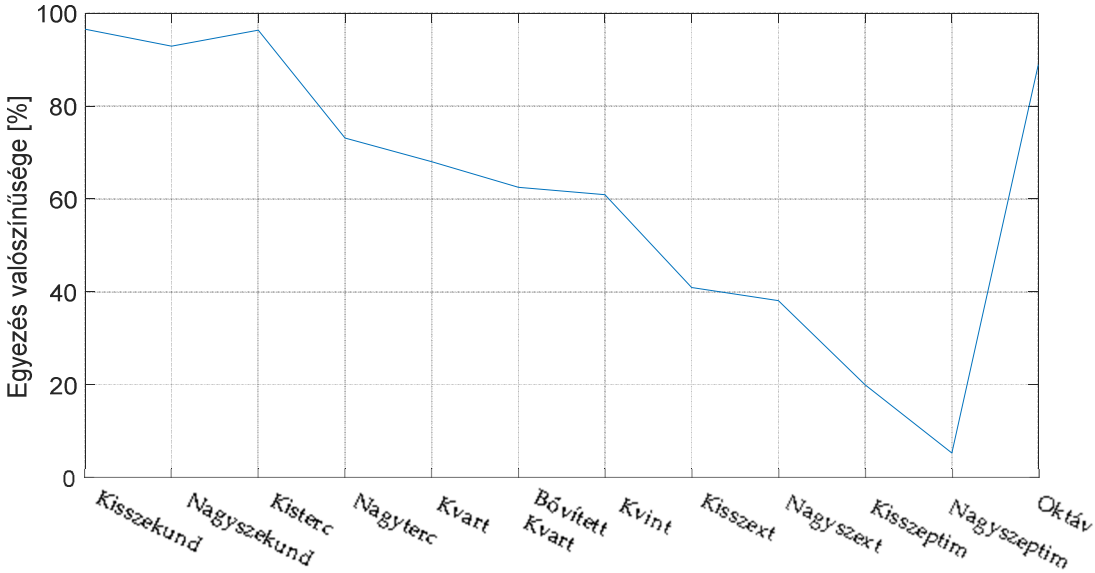
A hangközablakok használata első ránézésre megoldotta a hangkitörléses módszer problémáját, vagyis abban az esetben, amikor valóban oktáv szólal meg, akkor az esetek több, mint 90%-ban oktávnak ismeri fel. Ezek után a legelső teszteket kisebb hangköztávolságokkal (kisszekundtól-kvintig) végeztem, ahol a hangkitörléses módszerhez hasonló pontossággal – ami 90% feletti pontosságot jelent – találta el a hangmagasságot az algoritmus. A hangköz távolság növelésével viszont egy eddig ritkán tapasztalt hiba erősödött fel. Ezt a jelenséget hangközfordításnak szokás nevezni.



4-6. ábra — Hangközblakok használata (c4 – dis4 hangegyüttes)

Fizikai-akusztikai értelemben a hangközökre a bennük szereplő hangok rezgésszámainak hányadosa jellemző, pl. kvint = 3/2, kvart = 4/3, oktáv = 2/1. Hangközök összetétele esetén ezek az arányszámok összeszorzódnak: kvint + kvart = oktáv; 3/2 * 4/3 = 2/1. A kvint és a kvart egymást oktávra egészíti ki. 3/2 * 4/3 = 2. Ezért az egyik a másikból meghatározható úgy is, mint a reciprok kétszerese. Hasonló viszonyban áll egymással a nagyszekund és a kisszeptim; a nagyterc és a kisszext; a nagyszext és a kisterc; a nagyszeptim és a kisszekund [Budó 1997].

Hangközfordításnak nevezzük azt a műveletet, melynek során a hangköz alsó hangját egy oktávval feljebb, vagy a felsőt egy oktávval lejjebb helyezzük. A műveletből adódik, hogy egy hangköz és a megfordítása pontosan egy oktávra egészítik ki egymást.



4-7. ábra — Hangközablakok módszer pontossága százalékban

A hangközfordítás során prímből oktáv, szekundból szeptim, tercből szext, kvartból kvint lesz, és viszont. Tiszta hangköznek a megfordítása is tiszta, míg kis hangközből nagy hangköz, szűkítettből pedig bővített lesz, és fordítva.

Hangközablakokat alkalmazva a hangközfordítás lépett fel, mint egy újabb hiba, különösen a növekvő hangköztávolságok esetében (kiszext, nagyszext, kisszeptim, nagyszeptim). Leggyakrabban úgy keletkezett, hogy a hangköz alsó hangját egy oktávval feljebb érzékelte, így ami eredetileg szext volt, azt tercnek, ami pedig szeptim volt, azt pedig szekundnak detektálta (4-7. ábra). Ez a hiba akkora mértékű, hogy a korábban felsorolt hangközöket (szext és szeptim) 10 alkalomból mindössze 4-szer, de egyeseket egyszer sem találta meg helyesen. Mindemellett ugyanezeknél a hangközöknél fellépett egy kis mértékben az fésűs szűrők módszerének hibája is, azaz oktávnak érzékelt néhány olyan hangközt, ami valójában nem volt az.

4.5. Döntés egyértelműsége

Még mielőtt teljesen elvettem volna ezt a módszert, megvizsgáltam az algoritmus döntésének bizonytalanságát, vagyis hogy mennyire volt egyértelmű, hogy mely hangközt nevezi ki a megtalált hangköznek. Ezeket a döntéseket (eddig minden felsorolt módszer esetében) a hangközablakok segítségével számolt összegek alapján hoztam. Eltároltam a detektált hangköz energiáját (a legnagyobb energiával rendelkező hangköz), és a második legnagyobb energiával rendelkező hangközét, amely az esetek nagy többségében egybeesett a valóságban megszólaltatott hangközzel. Így minden egyes hangmagasság detektálás után maradt két energiaérték, melyeknek vettem a hányadosát. Ezek az aránypárok a hangközablakos módszert alkalmazva az esetek túlnyomó részében 1 közeli értéket vettek. Ez azt jelenti, hogy ugyanarra a hangközre két különböző szűrővel számolt energia gyakorlatilag megegyezik, így nem tudunk egyértelműen döntést hozni.

4.6. Eddig ismertetett módszerek értékelése

Habár a hangköz szerinti keresés első ránézésre jó eredményeket hozott, és bár maga a hangközfordítás nem számítana óriási hibának, mégis úgy döntöttem, hogy muszáj elvetni ezt az algoritmust, a döntési bizonytalansága miatt. Már-már szinte véletlenszerűen választ a két legnagyobb energiájú hangköz közül, a közel egyenlő értékekből adódóan. A véletlenszerű döntéshozatal miatt pedig nem lehet tovább építkezni erre az algoritmusra.

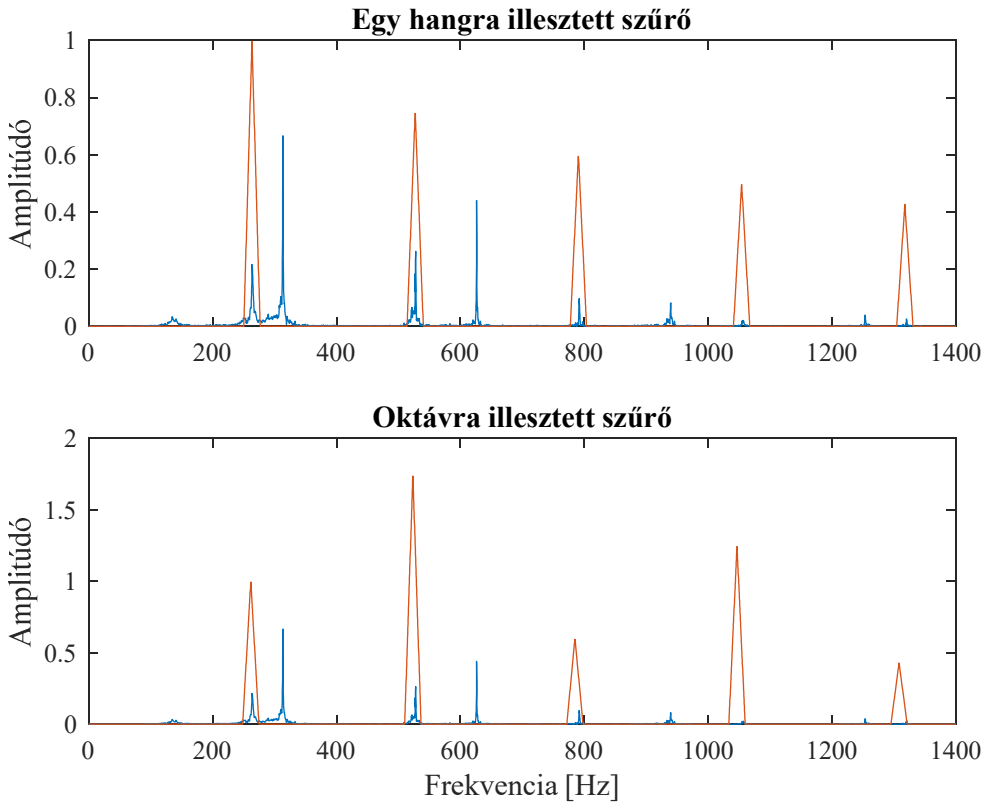
A fésűs szűrők módszere hiába volt eddig a futási időben a legjobb, és találta meg minden esetben a tiszta hangközöket, ha azon kívül nem tudott felmutatni jó eredményeket.

A hangkitörléses módszer adta eddig a legpontosabb eredményeket, leszámítva egyetlen hangközt, az oktávot. Ennek okát már korábban említve, az első számú megtaláltnak kinevezett hang törlésekor adatvesztés lép fel, hiszen olyan harmonikusok törlődnek, amelyek a másik hangnak is részét képezik, vagy esetleg teljesen el is tűntetem ezzel a második számú hangot. Mivel egy hangköz kivételével elég pontosan működik ez az algoritmus, ezért a következőkben ezt fogom továbbfejleszteni.

4.7. Második módszer – Oktáv probléma kiküszöbölése

A második módszer egyetlen nagy problémája, hogy az egyes számú hang kitörlésekor annak oktávja is gyakorlatilag törlődik, ha meg volt az adott időpillanatban szólaltatva. Tehát az a cél, hogy meg tudjuk állapítani az egyes hangközökről, hogy oktáv-e. Így csak akkor használnánk a hangkitörléses módszert, amikor tudjuk az adott hangegyüttesről, hogy nem oktáv.

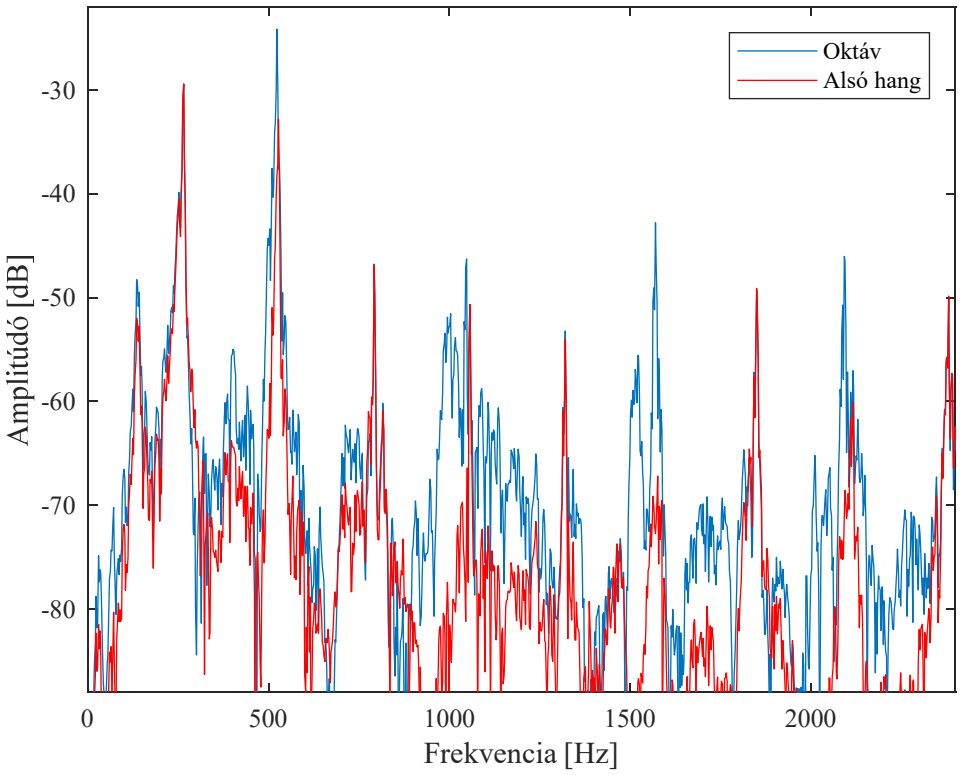
Habár a hangközablakok használatát elvetettem, mint önálló megoldást a hangmagasság detektálására, elkezdtem tesztelni, hogy alkalmas-e oktávok felismerésére. Kétféle fésűs szűrő segítségével vizsgálom meg az egyes hangokat: Az egyik egy csökkenő magasságú háromszögablakokból álló szűrő, a másik egy olyan, az előbbihez hasonló szűrő, melyben minden második háromszög csúcsa magasabban van, mint az előtte lévőé (4-8. ábra).



4-8. ábra — Egy hangra és oktávra illesztett szűrő

A továbbiakban az akusztikus gitár hangtartományában az összes különálló hangot és az összes hangközt a kisszekundtól az oktávig vizsgáltam e két fajta szűrőt használva, és tároltam el ennek megfelelően mindegyikhez két-két energiaértéket. Egy feladata van

ennek az algoritmusnak, mégpedig az, hogy eldöntse, hogy melyik szűrő segítségével számoltunk jelentősen nagyobb energiaértéket. Fontos, hogy jelentősen nagyobb legyen az egyik energiaérték, hiszen enélkül bizonytalan, már- már véletlenszerű a döntésünk.
Sajnos ez nem teljesült, újra abba a problémába ütköztem, hogy kicsi volt az energiakülönbség a két fésűs szűrőből számolva, így nem tudunk egyértelmű döntést hozni. Így más módszert kell keresni az oktávok detektálására.
Ahhoz, hogy algoritmus tudjak írni erre a problémára, először megfigyeltem egy-egy oktáv spektrumát, hogy miben tér el annak képe, mintha csak egy hang szólalna meg, vagy esetleg egy másik hangköz (4-9. ábra).



4-9. ábra — Oktáv és annak alsó hangjának spektrum képe

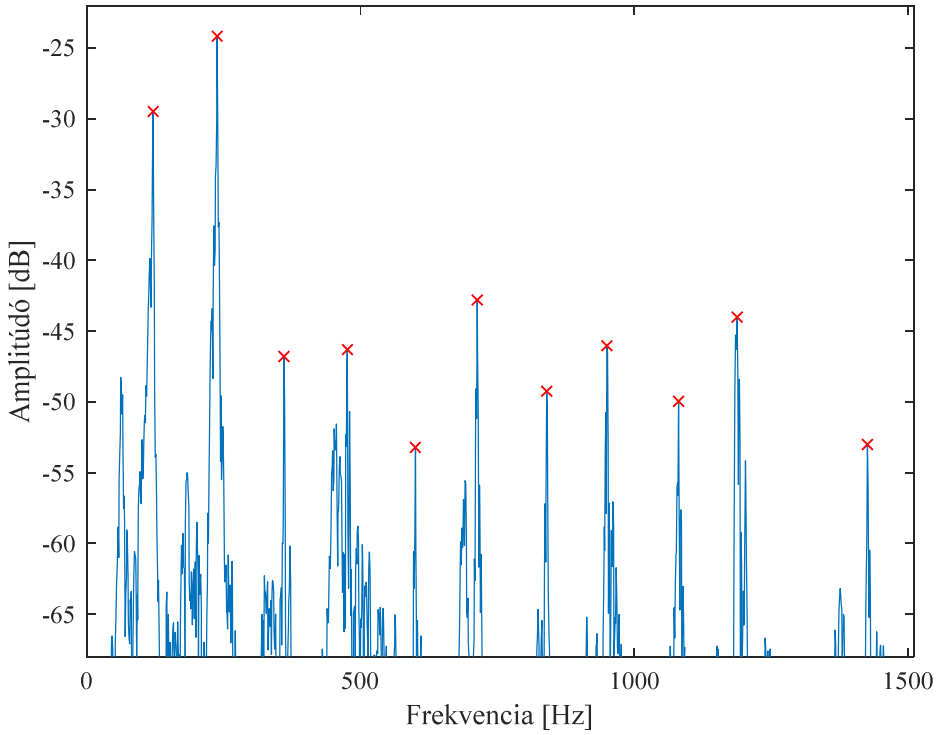
Ebben az esetben decibelben vizsgálom az értékeket, mert relatív sok harmonikust vizsgálok, illetve hasonlítok össze, és a döntésben ugyanakkora szerepe lesz a gyengébb komponenseknek is, mint az erősebbeknek.
Látható, hogy egyetlen hang esetében a harmónikusok amplitúdói csökkennek, ahogy növekszik a frekvencia. Az oktáv megszólaltatásakor is látható ez a csökkenés, viszont ez a csökkenés nem monoton, ugyanis a páros számú harmónikusoknak szinte kivétel nélkül magasabban van a csúcsuk, mint a körülöttük lévő harmonikusoké (4-9. ábra).

Emiatt próbálkoztam az eltérő magasságú háromszögablakok segítségével hangenergiát számolni. Mivel ez nem vezetett megoldásra, egy másik lehetőség az, hogy nem a harmónikusok energiáját vesszük figyelembe, hanem a harmonikusok csúcsait (4-10. ábra). Megkeresem egy meghatározott számú harmónikus csúcsainak helyét és értékét, majd minden csúcs értékét összehasonlítom a szomszédos csúcsokéval. Ha egy harmonikus csúcsa alapján az oktáv tulajdonságát fedezem fel benne, azaz ha egy páratlan számú harmonikus csúcsa kisebb mint az utána lévő harmonikus csúcsa, vagy ha egy páros számú harmonikus csúcsa nagyobb, mint az utána lévő harmonikus csúcsa, akkor egy változó segítségével eltárolom ezt az eredményt, majd megszámolom, hány harmónikus alapján ítélte azt az algoritmus, hogy ez egy oktáv. Ezek után összevetem a kapott számot az összesen vizsgált harmónikusok számával, vagyis hogy hány harmónikus alapján ítéltük volna a hangra vagy hangegyüttesre a vizsgáltak közül, hogy az egy oktáv-e. Ehhez szükséges volt meghatározni egy határt, hogy hány harmónikus oktávnak ítélése után mondhatom valóban a hangra, hogy az egy oktáv.

E határ meghatározásához létrehoztam egy tesztrendszert, mely végigfut az összes hangközön a prímtől az oktávig az akusztikus gitár hangtartományával, és feljegyzem minden hang és hangegyüttes esetében, hogy hányszor ítélte oktávnak a vizsgált harmónikusokból. A tesztben minden esetben az első 15 harmónikust vizsgáltam, minden egyes harmónikust (párost és páratlant egyaránt) a vele szomszédos, eggyel felette elhelyezkedő harmonikussal vetettem össze, így egy hang maximálisan 14 pontot kaphat arra a megállapításra, hogy oktávról beszélünk.

Azt a bizonyos határt 12 pontnál húzta meg a teszt, vagyis ha egy hang legalább 12 pontot kapott a 14-ből (azaz a vizsgált harmónikusok kb 85%-a alapján oktávnak ítélte), akkor oktávnak minősítjük. Ez volt az a határ az eredmények alapján, amely az egyedül megszólaló hangokat (prímeket) sosem minősítette oktávnak, kisszekundtól nagyszeptimig pedig csak nagyon kis százalékban.

Magán az oktávokon végzett teszt értelmezése kicsit összetettebb. Összesen 19 különböző oktávot vizsgáltam meg a tesztelés során, és ebből 5 alkalom kivételével az

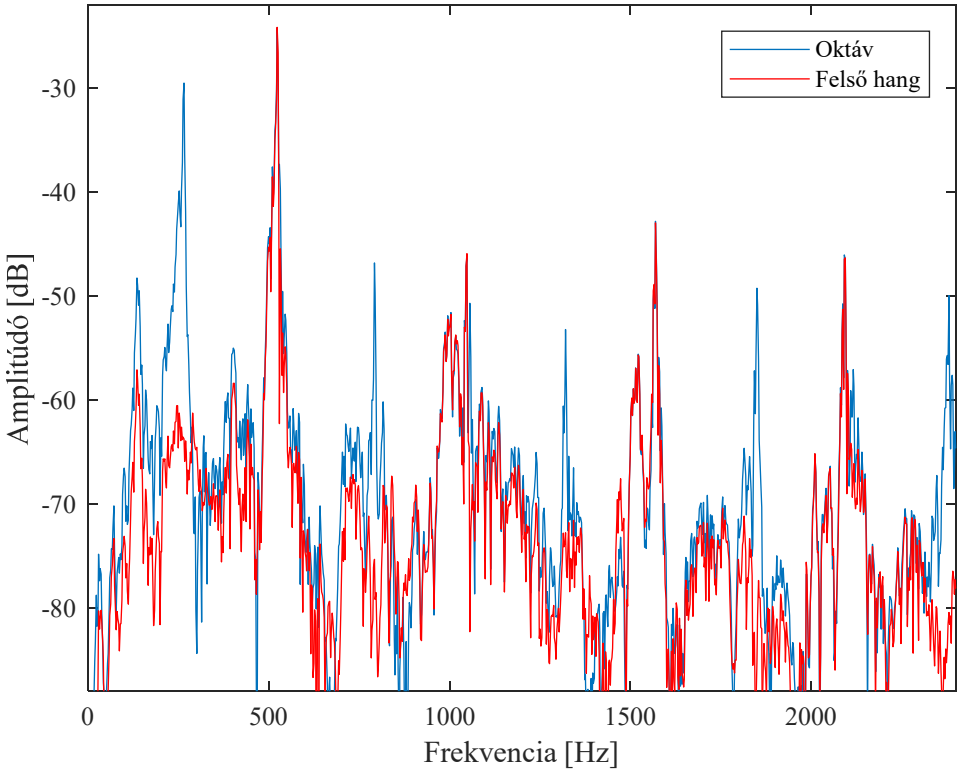


4-10. ábra — Egy oktáv spektruma és harmónikusainak csúcsai

eredmény az lett, hogy nem oktávot kapott a program a bemenetén. Ez az eredmény elsőre meglehetősen rossznak tűnik, de érdemes jobban megvizsgálni. A fésűs szűrők módszerének első felét is lefuttattam e teszt alatt, vagyis megkerestem a legnagyobb energiával rendelkező hangot és eltároltam annak midi kódját, és minden bemeneti oktáv alsó hangjának midi kódját. Így felfedeztem egy összefüggést, mely szerint amikor az oktávot nem ismerte fel oktávnak, akkor magát a hangmagasságot sem találta el helyesen. Pontosabban nem az alsó hangot találta meg az oktávnak, hanem a felsőt. Ebből adódik, hogy az oktávdetektálásnál nem az alsó hang alapharmónikusától kezdtem el vizsgálni a spektrumot, hanem a felső hang alapharmónikusától, ami az alsó hang első felharmónikusával egyezik meg. Tehát a spektrumot pontosan egy harmónikussal elcsúszva vizsgáltam, így minden páratlan harmonikus maximuma volt nagyobb mint a párosaké, az algoritmus pedig, pont ennek az ellenkezőjét akarta bebizonyítani, így tehát nem csoda, hogy az lett az eredmény ezeknél az eseteknél, hogy a bemeneti hangfájlon nem egy oktáv szólalt meg. Mindezzel együtt felmerülhet a kérdés, hogy problémát jelent-e ez a „hiba”? A válasz az, hogy nem feltétlenül. A második módszer esetében azért nem találta meg az oktávot egy bizonyos százalékban, mert az alsó hangot találta meg a

legnagyobb energiájú hangnak, amit így kijelölt az első számú megtalált hangnak, melynek összes harmonikusát kitörölte a spektrumból a második lépésben.

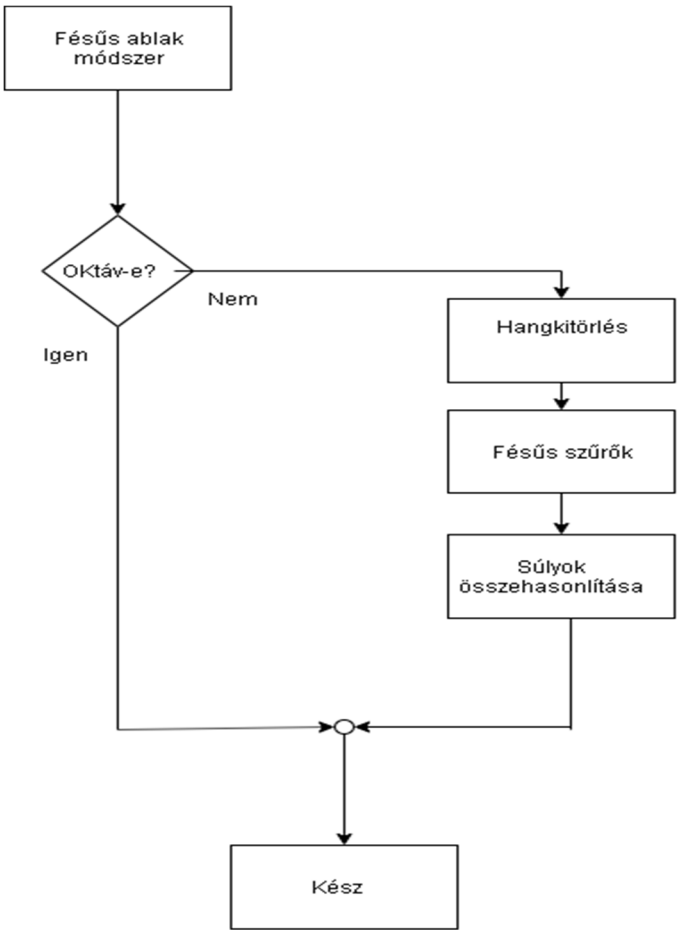
Ezzel a felette egy oktávval lévő hangot is kitörölte, mert annak harmónikusai egybeesnek az alatta egy oktávval elhelyezkedő hang egyes felharmónikusaival. Ezt a problémát ez az „oktáv teszt” megoldja, hiszen ha az alsó hangot találja erősebbnek, akkor felismeri, hogy az valóban egy oktáv. Ezért nem fog végigfutni a hangkitörléses módszernek nevezett algoritmuson, hiszen így a hangközt és a hangmagasságát is ismerjük.



4-11. ábra — Oktáv és annak felső hangjának spektrum képe

Ha a felső hangot találja meg erősebbnek, akkor egyszer sem lesz a teszt eredménye oktáv (4-11. ábra). Tehát elkezd futni a kitörléses módszer, azaz a második módszer. Már megtalálta az oktáv felső hangját, ami eddig helyes, hiszen valóban megszólaltatták. Ezt követően kitörli ennek harmónikusait, majd elkezdi keresni a megváltoztatott spektrumban megint a legnagyobb energiájú hangot. A magasabb hang harmónikusaival csak minden második harmónikust tüntetünk el az alsó hangból, annak alapharmónikusát is érintetlenül hagyja. Így a kitörlés után is marad elég információ a hangból, amely elég ahhoz, hogy az algoritmus megtalálja.

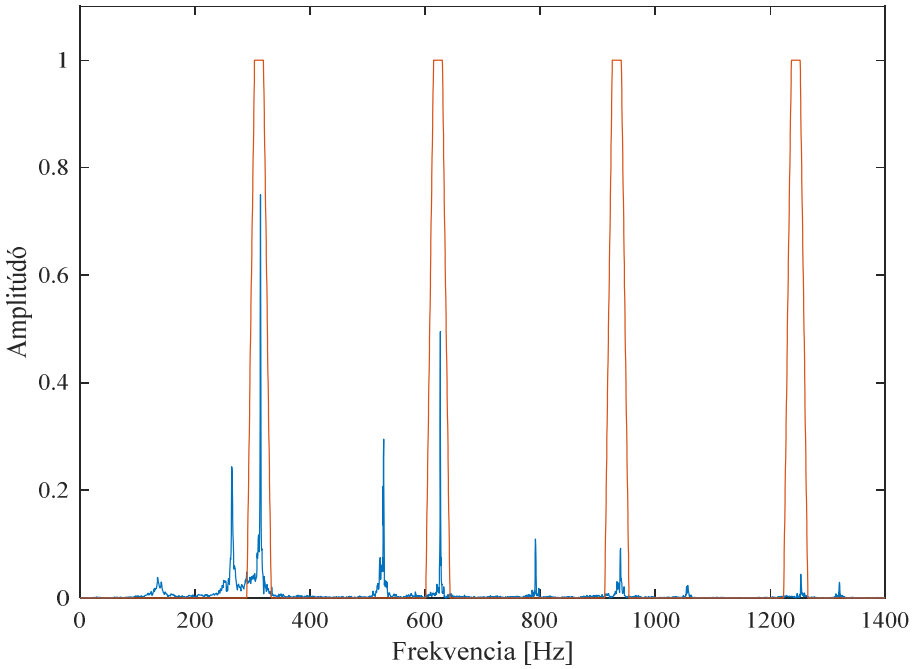
Mindezek után a következőképpen építettem fel a teljes algoritmust: A fésűs szűrők módszerével (melyet 4.1 fejezetben tárgyaltam) meghatározom az 1. számú felismert hangot, és az ahhoz tartozó összeget eltárolom. Majd e hang alapharmonikusából kiindulva megvizsgálom, hogy oktáv-e, ha igen, azzal egyszerre megállapítottam, hogy két hang szólalt meg együtt, és hogy a kettő között 12 félhangnyi távolság van. Ha az „oktáv teszt” szerint ez nem egy oktáv, akkor jön a hangkitörléses módszer, azaz az első számú hang harmonikusait kivonom a spektrumból, majd fésűs szűrők segítségével újra megkeresem azt a hangot, melyhez a legnagyobb összeg tartozik. Így jelenleg ugyanahhoz az időponthoz két összeget tároltam el. Célszerűen veszem ezek hányadosát (a második összeget elosztom az elsővel). Ha ez a hányados 1 közeli értéket mutat, tehát a két érték hasonló nagyságú, akkor tudjuk, hogy két hang szólalt meg ebben az időpontban egyszerre, és eltárolom a második összeghez tartozó hang midi kódját. Azonban ha ez a hányados a nullához közelít, akkor az algoritmus azt mondja, hogy csak egy hang szólalt meg, melyet még a program elején a fésűs szűrők módszerével határoztam meg (4-12. ábra).



4-12. ábra — Hangmagasság detektálás folyamatábrája

4.8. Apró módosítások

Van pár dolog, ami magán az algoritmus lefolyásán nem módosít, mégis jelentősen megváltoztathatja a program kimenetét. Erre egy példa, amit én is változtatgattam a félév során, a fésűs szűrők ablakainak alakja vagy azok ablakainak mérete. Különböző formákat tesztelgettem (téglalap, háromszög, trapéz), hogy mely ablakokkal kapom meg a lehető legpontosabb eredményt. Elsőként téglalapokból álló fésűs szűrőket használtam, ezt volt a legegyszerűbb előállítani. Azokon a helyeken, ahol keressük a harmonikusokat, ott e szűrő értéke mindig 1, egyébként 0. Maguk a harmonikusok körül zajos általában a spektrum, és ezt a zajt ugyanakkora értékkel súlyozza, mint a nekünk kellő információt. Ebből az okból kifolyólag váltottam le a téglalapot háromszögekre. A célom ezzel az volt, hogy a háromszögek csúcsait próbáltam illeszteni, a harmonikusok csúcsaihoz. Így a legerősebb súllyal véve az adott harmonikus frekvenciájánál lévő energiát, és egyre jobban elnyomja, minél távolabb helyezkedik el ettől a frekvenciától. A probléma ezzel csupán annyi volt – például azért, mert a hangszer nincs tökéletesen felhangolva- , hogy

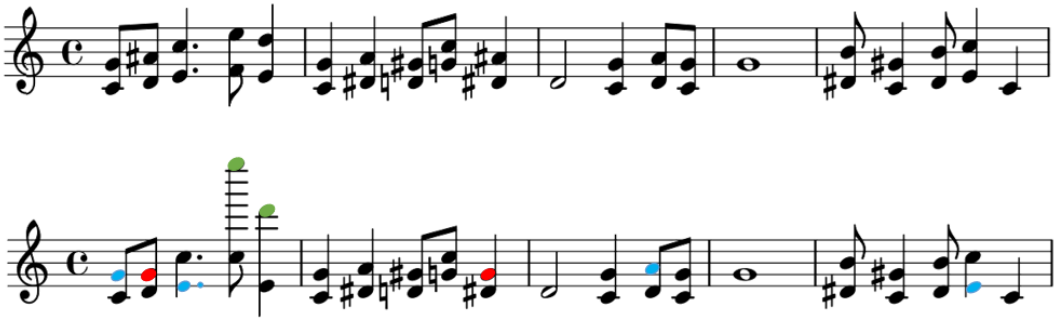


4-13. ábra — Trapéz ablak háromszögablak helyett

nem mindig esett egybe az ablak és a harmonikus csúcsa, néhol már teljesen elnyomta a szűrő az adott harmonikust. Így döntöttem egy olyan ablak mellett, mely nem csak egyetlen pontban enged át teljes mértékben, hanem egy kisebb sávban, de nem éles annyira a választóvonal, mint a téglalap esetében (4-13. ábra). A trapéz formájú ablakok használatával 1-2%-kal nőtt a hangmagasság detektálás pontossága.

5. Kiértékelés

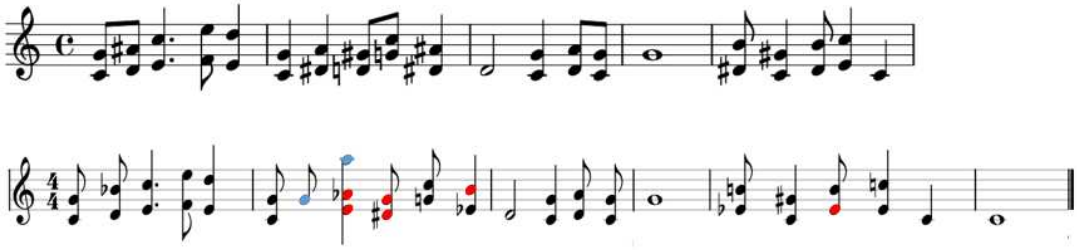
Ahhoz, hogy tesztelhessem az algoritmust, olyan zenerészleteket érdemes alkalmaznom, melyeknek ismerem a kottaképét, hogy legyen mivel összehasonlítanom az végeredményt. Ezeket a hangmintákat ideális körülmények között az FL Studio segítségével generáltam.



5-1. ábra — Eredeti és az algoritmusom által generált kottakép

Az 5-1. ábrán látható egy példa: felül az eredeti kotta, alul pedig az algoritmusom által generált kotta képe látható. A hangok elhelyezkedését, a hangértékeket és tempót illetően egyszer sem tévesztett e bemeneti tesztfájl esetében. Ezzel szemben a hangmagasság detektálása már nem volt ennyire tökéletes. Az 5-1. ábrán az eltéréseket különböző színekkel jelöltem a generált kottában. Kétszer detektált hibás hangot (piros szín), a másik két hangtévesztés esetében a keresendő hangnak (amit meg kellett volna találjon) az oktávját találja meg (zöld szín), vagy a felette két oktávval lévő hangot tehát oktávtévesztésről van szó. A maradék fennálló hibák az együttesen megszólaló hangok számának a becsléséből adódott (kék szín). Néhány helyen, ahol tudjuk, hogy két hang szerepel, csak egy hangot érzékelt az algoritmus, de láthatóan nem olyan gyakori a hiba, hogy problémát jelentsen.

Ezt a bemeneti tesztfájlt a már korábban a bevezetőben említett *AnthemScore* segítségével is feldolgoztam, hogy lássam milyen eredményt ad az a program, melyet vezető szoftverként tartanak számon e területen [AnthemScore].



5-2. ábra — Eredeti és az AnthemScore által generált kottakép

Ismét felül látható az eredeti kottakép, alatta pedig az, amit szoftver generált (5-2. ábra). A színek ugyanazokat a hibákat jelölik, mint az 5-1. ábrán. Megfigyelhetjük, hogy az *AnthemScore* sem teljes mértékben tökéletes hangmagasság detektálás terén. Esetében is előfordul, hogy hangot téveszt, vagy hogy rosszul határozza meg az egy időben megszólaló hangok számát. Ennek ellenére figyelembe kell venni azt is, hogy az általam létrehozott algoritmus csak két szólamig tud helyesen működni, attól többre nincs felkészítve, míg az *AnthemScore* nincs ilyen határok közé szorítva.

	OKTÁV HIBA	HIÁNYZÓ VAGY TÖBB HANG	HANGTÉVESZTÉS
MÉZGA CSALÁD	0%	15%	6%
LET IT GO	32%	21%	0%
A THOUSAND YEARS	15%	30%	7%
GYAKORLATOK	0%	11%	22%
CSUKÁS ISTVÁN-DALOCSKA	4%	33%	29%
BOCI BOCI	56%	22%	7%
CSIGA BIGA	14%	7%	14%
LÁTTÁL-E MÁR VALAHA?	4%	37%	12%
HINTA PALINTA	0%	5%	0%
MEGY A KOCSI	4%	15%	11%

5-1. táblázat — Bekövetkezett hibák adott gitáron játszott dalok esetén

Több dalra is lefuttattam tesztelés végett az általam létrehozott algoritmust, és feljegyeztem az fentebb bemutatott háromféle hibalehetőség (oktáv tévesztés, több vagy kevesebb hang detektálása, hangtévesztés) bekövetkezését százalékban kifejezve. Az 5-1. táblázatban gitár, az 5-2. táblázatban pedig zongora által megszólaltatott zenék kottázása esetén fellépő hibák arányai láthatók. Téves hang észlelése, ahogy reméltem, nem volt túl gyakori (a tesztelt dallamokban a legrosszabb esetben is a hangok kevesebb, mint a 30%-át detektálta helytelenül). A zongorára is hasonlóan jó eredményeket kaptam, mint a gitár dallamokra, néhol még pontosabb is volt zongora esetében. Bár ez lehet annak a következménye, hogy az FL Studio a gitár esetében zajosabb hangmintákkal dolgozik.

	OKTÁV HIBA	HIÁNYZÓ VAGY TÖBB HANG	HANGTÉVESZTÉS
MÉZGA CSALÁD	0%	58%	3%
LET IT GO	0%	12%	0%
A THOUSAND YEARS	15%	30%	0%
GYAKORLATOK	0%	17%	33%
CSUKÁS ISTVÁN-DALOCSKA	0%	4%	29%
BOCI BOCI	0%	11%	7%
CSIGA BIGA	11%	34%	0%
LÁTTÁL-E MÁR VALAHA?	0%	0%	0%
HINTA PALINTA	0%	0%	5%
MEGY A KOCSI	0%	2%	2%

5-2. táblázat — Bekövetkezett hibák adott zongorán játszott dalok esetén

Alapvetően szintetikusan teszteltem, de megfigyeltem pár jó minőségű akusztikus felvétel esetén is az eredményeket. Ebben az esetben az egyszerre megszólaló hangok számának tévesztése nagyjából kétszerese volt, mint amikor az FL Studio segítségével generáltam a hangmintákat. Maga a hangtévesztés előfordulása nem növekedett jelentősen. Az AnthemScore szoftver eredményének pontosságát nem befolyásolta jelentősen az, hogy szintetikusan előállított vagy akusztikus hangfelvételeket használtam.

6. Összegfogalalás

A dolgozat elkészítése kapcsán megismerkedtem az automatikus kottázás fő irányvonalaival, és létrehoztam egy olyan egyszerű programot, mely bizonyos korlátok között jó pontossággal felismeri a zenei hangokat és a zenék ritmusát. E két paraméter, amelynek ismerete feltétlenül szükséges egy kottakép felrajzolásához. Többféle módszert próbáltam ki ütem-illetve hangmagasságdetektálás terén MATLAB segítségével. Többnyire a hang energiáját számoltam különböző szűrők (általában fésűs szűrők) segítségével, vagy éppen a csúcsértékeket vizsgáltam bizonyos jelekben. Így több eljárást hasonlítottam össze, hogy kiválaszthassam a legideálisabbat a végső algoritmushoz.
A felállított követelményeket az elkészült program sikeresen teljesítette, tehát képes megfelelő pontossággal kottát felírni egyszerűbb, maximum kétszólamú dallamok alapján. A fejlesztés során további feladatot jelentett a feldolgozás felgyorsítása. Néhány esetben azért vetettem el egy-egy algoritmust, mert túl nagy számításigénye volt, jelentősen megnövelte a futási időt, ami egy hosszabb zenei darab feldolgozásakor nagyon sokat számít.
Az általam készített algoritmus futtatásához nincs szükség semmilyen speciális eszközre, csak egy napjainkban minden háztartásban megtalálható számítógépre. Bemenetére egyszerűen egy mp3 kiterjesztésű fájlt vár. Az algoritmus ebből a fájlból legenerál egy egyszerű kódot, melyből a *LilyBin* ingyenes online kottaszerkesztő már közvetlenül fel tudja rajzolni a kívánt kottaképet.

Az algoritmus továbbfejlesztési lehetőségei:

Korábban említettem, hogy bizonyos feltételek mellett közel jó eredményt kapunk e algoritmussal. Feltételezzük, hogy a bemenő dallam legfeljebb kétszólamú, míg más hasonló programok (mint például a bemutatott *AnthemScore*) megengednek több szólamot is.

1. Így tehát ebből rögtön adódik az első továbbfejlesztési lehetőség, azaz további szólamokat hozzáadva is képes legyen kellő pontossággal felismerni a hangegyütteseket.

2. A továbbiakban a hangnemfelismeréssel is lehetne foglalkozni (a már felismert hangok hangsorokra való illesztésével meghatározható az egyes dallamok hangneme).

3. Korábban a hangértékek meghatározásánál feltételeztem, hogy egy hang addig van kitartva, míg egy másik hang nem érkezik. Ez azonban a vaslóságban ritkán van így, hiszen a zene nem csak hangokból állnak, hanem hangokból és szünetekből. Időtartománybeli képet tovább vizsgálva, meghatározhatnám a jövőben az egyes hangok hosszát, vagyis hogy hol vannak azok a pontok, ahol az adott hang elhalkul. Ezzel kiderülne, ha korábban például egy félhangot és egy félhangnyi szünetet egy egészhangnak detektáltam.

4. További tesztek elvégzése valós, gyengébb minőségű hangfelvételeken, és a program módosítása úgy, hogy azok kottaképeit is megfelelő pontossággal tudja meghatározni.

7. Melléklet ismertetése

Automatic_transcription.m: az elkészült szoftver futtatható verziója MATLAB környezetben.

midi.dat: MIDI táblázatot tartalmazza, 1. oszlopban midi kódok, 2. oszlopban a hozzájuk tartozó alapfrekvencia helyezkedik el.

sheet.dat és *text.dat*: LilyBin használatához szükséges szintaktikai elemeket tartalmazzák.

README.txt: A melléklet rövid összefoglaló leírását tartalmazza.

Matlab kiegészítő fájlok: A Matlabban elkészített algoritmusok által használt funkciók fájljai _funcion.m végződéssel rendelkeznek. Ezek szükségesek a kódok helyes működéséhez.

Hangfájlok:

thousand_zongora.mp3, thousand_gitar.mp3, thousand_akusztikus.mp3, lattal_e_mar_valaha_zongora.mp3, lattal_e_mar_valaha_gitar.mp3, let_it_go_zongora.mp3, let_it_go_zongora.mp3, csukas_istvan_dalocska_zongora.mp3, csukas_istvan_dalocska_gitar.mp3, dal.mp3

8. Irodalomjegyzék

[AnthemScore] **Audio to Sheet Music With Machine Learning**, Utolsó letöltés: 2019.12.12.
https://www.lunaverus.com/

[Antony 2009] **Antony Schutz, Dirk Slock** — **Periodic Signal modelling for the octave problem in music transcription,** DSP 2009, 16th International Conference on Digital Signal Processing, July 05-07, 2009, Santorini, Greece

[Balázs 2013] **Balázs János** — **Beat Detection and Correction For Djing Applications,** Budapest University of Technology and Economics, Msc thesis – 2013.
http://dsp.mit.bme.hu/userfiles/diploma/balazsdiploma13.pdf

[Budó 1997] **Dr. Budó Ágoston** — **Kísérleti Fizika, I. Kötet (Mechanika, Hangtan, Hőtan)**, Nemzeti Tankönyvkiadó Rt. 1997

[Horváth 2017] **Horváth Gergely** — **Ritmusérzék-fejlesztő alkalmazás fejlesztése**, Szakdolgozat, BME Méréstechnikai és Információs Tanszék, Budapest, 2017.
http://dsp.mit.bme.hu/userfiles/szakdolgozat/horvathgszakdolgozat17.pdf

[Kanizsai 2012] **Kanizsai Rita** — **Zenei fogalmak és rendszerek a matematika nyelvén**, BSc szakdolgozat, ELTE Numerikus Analízis Tanszék**,** Budapest, 2012.
https://web.cs.elte.hu/blobs/diplomamunkak/bsc_mattan/2012/kanizsai_rita.pdf

[Kesztler 1959] **Kesztler L. Dr. — Zenei alapismeretek - Iskolai és magánhasználatra.** Zeneműkiadó Vállalat, Budapest, 1959. 10-11 181-182

[Kotta] https://www.zenci.hu/szocikk/kotta, Utolsó letöltés: 2019.12.12.

[Lipovszki 2012] **Dr. Lipovszki György** — **Jelfeldolgozás és számítógépes irányítás**, EDUTUS Főiskola – 2012.
https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2010-0017_36_jelfeldolgozas_es_szamitogepes_iranyitas/adatok.html

[Löblin 1982] **Löblin Judit** — **A hangjegyírás a kezdetektől a tipográfiáig**, Magyar Grafika, 1982. 1-2. sz.
http://mek.niif.hu/00200/00207/html/n05.htm

[Szigetvári 2014] **Szigetvári Andrea, Horváth Balázs** — **Bevezetés a zenei informatikába -Hangspektrumok osztályai**, Budapesti Kommunikációs és Üzleti Főiskola, Typotex Kiadó, Budapest 2014,
https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0010_szigetvari_bevezetes/ch03s02.html.

[Williams 2013] **Victoria Williams** — **Time Signatures - 4/4 or C?**, 2013.Dezember 11., Utolsó letöltés: 2019.12.12.
https://www.mymusictheory.com/learn-music-theory/reference/347-time-signatures-4-4-or-c

[Zheng 2007] **Zheng Guibun és Liu Sheng** — **Automatic Transcription Method for Polyphonic Music Based on Adaptive Comb Filter and Neural Network**, IEEE International Conference on Mechatronics and Automation – 2007.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4303965