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

Nemes Dániel

FENDER RHODES ELEKTROMOS ZONGORA MODELLEZÉSE MATLAB ÉS VST KÖRNYEZETBEN

KONZULENS

Dr. Firtha Gergely

BUDAPEST, 2020

SZAKDOLGOZAT FELADAT

Nemes Dániel
Villamosmérnök hallgató részére

Fender Rhodes elektromos zongora modellezése MATLAB és VST környezetben

Harold Rhodes zongoraművész nevéhez fűződik a széles körben elterjedt és számos zenei stílusban használt jellegzetes hangszínű elektromos zongora, a Rhodes piano kifejlesztése. A 20. század második felétől Rhodes társult Leo Fender hangszertervezővel, az általuk alkotott modellek manapság is töretlen népszerűségnek örvendenek.
A hangszer hangkeltési mechanizmusa abban tér el a hagyományos zongoráétól, hogy a billentyű lenyomására működésbe lépő kalapács nem egy húrt, hanem egy aszimmetrikus hangvillát szólaltat meg. Ezt a rezgést elektromos jellé alakítva, valamint egy túlvezérelt erősítőn átvezetve jön létre a hangszer jellegzetes hangja.

A hangszer fizikai modellezése összetett, jól publikált feladat, amely azonban valós idejű hangszintézishez közvetlenül nem alkalmazható.
Alternatív megoldásként, hallgató feladata a rendelkezésre álló zongora-hang mérési adatbázisának analízise alapján egyszerű matematikai modell felállítása a különböző hangmagasságú és leütés-erősségű zongorahangok leírására, valamint a hangszintézisre alkalmazható modell implementálása MATLAB és VST környezetekben.

A hallgató feladatának a következőkre kell kiterjednie:

- Ismerje meg a mérésalapú hangszintézis módszereit!
- Illesszen modellt MATLAB környezetben a rendelkezésre álló zongorahang mérési adatbázisra!
- Ismerje meg a szoftverhangszerek fejlesztésére alkalmazott VST fejlesztőkörnyezetet!
- Készítsen grafikus felhasználói felülettel rendelkező VST plugint a létrehozott elektromos zongora-modell valós idejű hangszintézisre képes implementációjához!

Tanszéki konzulens: Dr. Firtha Gergely, egy. adjunktus

Budapest, 2020. október 11.

Dr. Imre Sándor
egyetemi tanár
tanszékvezető

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Nemes Dániel**, 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, 2020. 12. 11.

...…….

Nemes Dániel

Összefoglaló

A Rhodes egyike a piacon elérhető leghíresebb és legközkedveltebb elektromos zongoráknak. Az elmúlt évtizedek során Leo Fender hangszerműhelye modellek széles választékát alkotta meg, melyek azóta digitális- és virtuális hangszerként is töretlen népszerűségnek örvendenek. A számítástechnika fejlődésének köszönhetően tehát jogos alkotói igény merült fel egyes hangszerek szoftveres reprezentációja iránt. Jelen szakdolgozat célja egy egyszerű matematikai modell megalkotása a rendelkezésre álló mérési adatbázis hangmintáinak illesztésével, majd ez alapján valós idejű hangszintézisre képes virtuális hangszer megvalósítása.

A dolgozat első fejezete ismerteti a Fender Rhodes történetét és működési mechanizmusát, a szoftveres implementáció indokoltságát, valamint az alkalmazott szintézistechnikát. Kitértem továbbá az alkalmazott hangszeres digitális interfész (MIDI) és a virtuális stúdió technológia (VST) szabványokra.

A második fejezetben a mért hangminták alapján meghatároztam tetszőleges billentyűleütés hangjának feltételezett matematikai formuláját. A mérési eredmények időbeli és spektrális burkolóinak illesztésével megbecsültem a modell minden szükséges tulajdonságát MATLAB környezetben.

A harmadik fejezet tárgyalja az előállított modellparaméterek alapján történő additív hangszintézis folyamatát.

A negyedik fejezetben ismertetem a szoftverhangszer megvalósítása során felhasznált VST fejlesztőkészletet, majd a C++ nyelvű implementáció lépéseit.

Az utolsó fejezetben az illesztett modell elhanyagolásainak és közelítéseinek értékeléséről, valamint egyes továbbfejlesztési lehetőségekről esik szó.

Abstract

The Rhodes piano is one of the most famous and popular electric pianos available on the market. Over the past decades Leo Fender's instrument manufacturing company has created a wide variety of models which have since been made into digital- and virtual instruments. Thus, due to the development of computer technology artists had a reasonable need for software representation of certain instruments. This thesis aims to identify a simple mathematical model by fitting analytic expressions to the sound samples of the available measurement database. Based on this, I implemented a virtual instrument capable of real-time sound synthesis.

The first chapter of my thesis describes the history of Fender Rhodes and its mechanism of action. I discussed the justification of software implementation, the applied synthesis strategy, MIDI and VST interface standards.

In chapter two I determined the assumed mathematical formula for the sound of any piano key based on the measurement samples. By fitting curves to the amplitude and spectral envelopes of these samples I estimated all the required parameters of the specified model in MATLAB environment.

The third chapter describes the process of additive sound synthesis based on the model parameters.

In the fourth chapter I explained the components of the VST software development kit and the steps taken during C ++ implementation.

The final chapter is about the estimates and approximations made in this work. I also mentioned some alternatives for further development.

1 Bevezetés

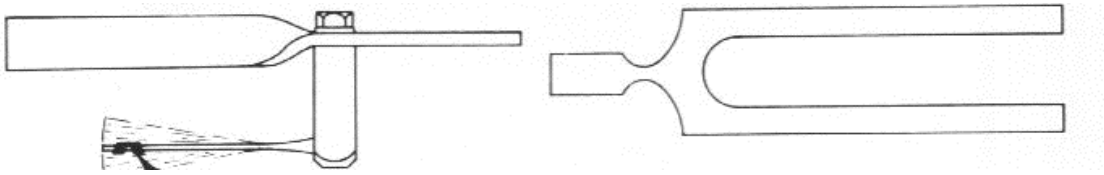
1.1 Történeti áttekintés

A Rhodes kifejlesztése az amerikai zongoratanár és zenész, Harold B. Rhodes nevéhez fűződik. A feltaláló a II. világháború idején a sebesült katonák felépülését segítette azzal, hogy zongorázni tanította őket. Gyors, egyszerű és szórakoztató tanítási módszerével lenyűgözte társait, ezért úgy döntött, hogy feleslegessé vált repülőgépalkatrészek felhasználásával kifejleszt egy két és fél oktávos elektromos zongorát ahhoz, hogy az ágyban fekvő, sérült katonák is gyakorolhassanak. A hangszer és a katonák zenei képzése annyira sikeres és népszerű lett, hogy a hadügyminisztérium kérésére a modell tervrajzai alapján megkezdték a gyártást. 1942 és 1945 között mintegy 125 000 darab készült, majd az 1950-es évektől Rhodes társult Leo Fender hangszertervezővel, hogy nagyobb modelleket is kifejlesszenek (1.1. ábra). Az általuk alkotott darabok a mai napig töretlen népszerűségnek örvendenek főként a jazz, a pop és a soul zenében [1].

1.1. ábra. A *Fender Rhodes Stage 73* modell [2]

1.2 A hangszer működése

A Rhodes működési mechanizmusa abban tér el a hagyományos zongoráétól, hogy a billentyű lenyomására működésbe lépő kalapács nem egy húrt, hanem egy aszimmetrikus hangvillát (nemzetközi nevén: „tine”-t) szólaltat meg (1.2. ábra). Ezt a rezgést elektromágneses hangszedők alakítják analóg elektromos jellé, majd az így kapott jelet egy túlvezérelt erősítőn átvezetve és egy hangszóróra kötve hallgathatjuk meg a hangszer jellegzetes hangját.



1.2. ábra. „Rhodes tine” hangolórugóval (balra) és hagyományos hangvilla (jobbra) [3]

A hangszerből 54, 73 és 88 billentyűs modelleket gyártottak, melyek teljesen polifonikusak voltak. Külső bemeneti forrásként csatlakoztatható hozzájuk lábkapcsoló (nemzetközi nevén: sustain pedal), kimenetüket pedig billentyűerősítőre vagy akár keverőpultra is köthetjük. Egyes típusokat tremoló-effekttel is elláttak [1].

1.3 Miért pont szoftverhangszer?

A digitális audio jelfeldolgozás fejlődése, a számítástechnika térnyerése, valamint a számítási kapacitás növekedése egyaránt különböző szoftverszintetizátorok térhódítását eredményezte. Ezek többnyire beépülő szoftvermodulként (pluginként), vagy dedikált hardveregység (pl.: DSP) vezérlőprogramjaként készülnek. Ennek eredményeképpen jelentős felhasználói igény jelentkezik virtuális hangszerekre, ugyanis áruk és hozzáférhetőségük sokkal szélesebb körben teszi lehetővé jó minőségű felvételek készítését, vagy akár a megvalósított modell valós idejű szintetizátorként való használatát. Ehhez nem kell egyéb, mint egy számítógép, egy MIDI-billentyűzet, egy megfelelő hangkártya, egy szoftverhangszerek fogadására képes gazdaprogram (hoszt) és természetesen a választott szoftverhangszer. A hoszt jellemzően egy digitális audio munkaállomás (nemzetközi nevén: Digital Audio Workstation, röviden: DAW), amely egy audio felvételre, szerkesztésre és tartalom-előállításra alkalmas szoftver.

Számos DAW alkalmazás és virtuális hangszer érhető el a piacon, köztük néhány igencsak kedvező áron, vagy akár ingyenesen is letölthető. A Rhodes-szintetizátorok egyik legnépszerűbb képviselőjét mutatja az 1.3. ábra.

1.3. ábra. *Arturia Stage-73 V*: az egyik piacvezető Rhodes-szintetizátor [4]

1.4 A mérésalapú hangszintézis

Hangszintézisnek nevezzük a hang elektronikus eszközökkel való létrehozását. Ennek során többnyire a jelek matematikai vagy fizikai leírásából indulunk ki, tehát olyan elektromos jelet állítunk elő, melynek feszültség-időfüggvénye megfeleltethető a hang hangnyomás-időfüggvényének. Digitális szintézis során olyan szintézistechnikára van szükség, amely valamilyen matematikai vagy fizikai modell formájában lehetővé teszi a kívánt hangtulajdonságok reprezentációját. A szintézis-stratégiák száma az első szintetizátor megjelenése óta folyamatosan bővül, módszereik pedig egyre komplexebbé, kifinomultabbá válnak [5].

-A Fender Rhodes elektromos zongora fizikai modellezése összetett, jól publikált feladat (lásd: [6] forrás), amely azonban valós idejű hangszintézishez közvetlenül nem alkalmazható. Alternatív megoldásként, jelen dolgozat célja egy matematikai modell előállítása a különböző leütéserősségű és hangmagasságú zongorahangok leírására. Ehhez a mérésalapú hangszintézis módszereit alkalmazzuk,

vagyis a modellünkhöz a [6] irodalomban mellékelt zongorahang-mérési adatbázist vesszük alapul. Ezek a hangminták kiváló minőségű felvételek, ezért feldolgozásuk során igyekszünk minél precízebben megközelíteni őket. A módszer előnye, hogy ezáltal bármilyen választott hangszer hangját nagy pontossággal állíthatjuk elő. Hátránya azonban, hogy kizárólag a mérés alapjául szolgáló hangszer hangja reprezentálható, így a módosítható paraméterek száma is korlátozott. A modellt MATLAB környezetben valósítottam meg.

1.5 A MIDI szabvány

A hangszeres digitális interfész (nemzetközi nevén: Musical Instrument Digital Interface, röviden: MIDI) a számítógépek és az elektronikus hangszerek közti kommunikáció műszaki szabványa. A technológia talán leggyakoribb alkalmazása a MIDI billentyűzetek és kontrollerek használata, ezekre részletesen a 4.1.2. fejezetben ben tértünk ki. Ezek az eszközök lényegében nem hangjelet, hanem MIDI-üzeneteket generálnak, amelyek a leütésnek csak bizonyos tulajdonságait tárolják, mint például a leütött hang magassága (nemzetközi nevén: pitch), a leütés erőssége (nemzetközi nevén: velocity), valamint egyes vezérlőesemények.

A MIDI-esemény dekódolásáért az ehhez használt DAW alkalmazás vagy szoftverhangszer felel, így hangszerek és hangszínek széles választékát szólaltathatjuk meg [7]. Az üzenet útját a hang megszólalásáig az 1.4. ábra szemlélteti.

1.4. ábra. A MIDI-esemény útja

1.6 A VST interfész

A virtuális stúdió technológia (nemzetközi nevén: Virtual Studio Technology, röviden: VST) a német Steinberg cég által kifejlesztett szoftver interfész, mely lehetővé teszi virtuális hangszerek és effektek DAW szoftverekbe való integrációját. A már-már szabványossá vált modulokat („VST-pluginokat”) három fő kategóriára oszthatjuk.

A *VST hangszerek* (nemzetközi nevén: VST Instruments, röviden: VSTi) célja egy adott hangszer hangjának utánzása (emulációja). A VSTi tehát hangot generál valós idejű vagy előre felvett bejövő jel, vagy MIDI-üzenetek hatására. Jelen dolgozat célja egy grafikus felhasználói felülettel rendelkező és MIDI-vezérelhető VSTi plugin megvalósítása a létrehozott elektromos zongora-modell valós idejű hangszintézisre képes implementációjához.

A *VST effektek* (nemzetközi nevén: VST Effects, röviden: VSTfx) célja nem hangok generálása, hanem a beérkező audio jel feldolgozása, esetleg egyes paramétereinek megváltoztatása a felhasználó által. A VSTfx szintén működhet tetszőleges bejövő hangjelre és MIDI-üzenetekre egyaránt. Egyes VST effektek a hardveres audio effektekkel azonos funkciókat látnak el (pl.: zengetők, torzítók), egyesek pedig a bemeneti hang monitorozását biztosítják, melynek során a felhasználó vizuális visszajelzést kaphat a jelről (pl.: spektrumanalizátorok, mérők).

A *VST MIDI effektek* MIDI-üzeneteket dolgoznak fel és lehetőséget biztosítanak egyes paraméterek megváltoztatására (pl.: leütéserősség-változtatás, transzponálás) [8].

A Steinberg Media Technologies 1996-ban adta ki az első VST specifikációt és a hozzátartozó fejlesztőkészletet (nemzetközi nevén: VST Software Development Kit, röviden: VST SDK). Ezzel egyidejűleg hozták forgalomba a Steinberg Cubase nevű DAW-szoftvert, amely már tartalmazta az első VST-pluginokat. 1999-ben a cég frissítette az interfészt a 2.0-s verzióra, amelynek egyik leglényegesebb kiegészítése volt, hogy lehetőséget nyújtott MIDI-üzenetek fogadására. Ez népszerűsítette az első szoftverszintetizátorok, samplerek és dobgépek fejlesztését [8].

2011-es megjelenésével lényeges újításokat hozott a 3.5-ös verzió. A cég megalkotta a *note expression* funkciót, amely átfogó információkat biztosít egyes leütések MIDI-eseményeihez kapcsolódóan. Ezeket az adatokat vezérlőeseményeknek nevezzük, céljuk pedig, hogy a MIDI-üzenetekkel egységet alkotva minden megszólaltatott hanghoz pontos és intuitív szerkesztési lehetőségeket, ezáltal természetesebb játékérzetet biztosítsanak [9]. A funkció biztosította lehetőségeket jelen dolgozat készítése során is kihasználtuk.

2 A modell megalkotása MATLAB környezetben

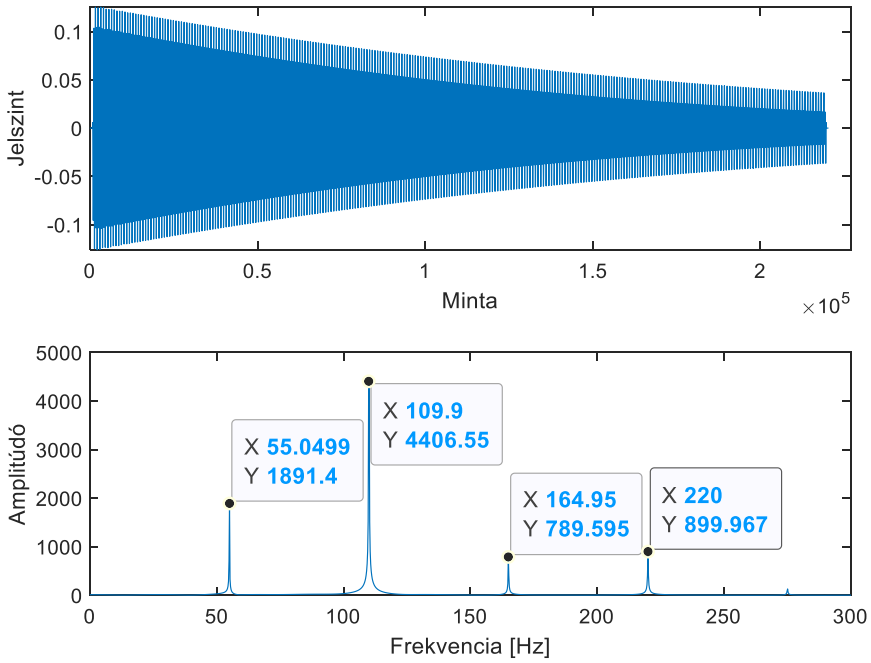
Jelen dolgozat első felében modellt illesztünk a [6] forrásban rendelkezésre álló zongorahang-mérési adatbázisra. A felvétel eredményeit a melléklet *samples* mappájában csatolt négy különböző *.mp3* fájl tartalmazza, melyek négy különböző hangmagasságú billentyű leütését tárolják egyenként öt különböző leütéserősségben. Célunk ennek megfelelően egy tetszőleges frekvenciájú és ütéserősségű zongorahang leírására alkalmas matematikai formula előállítása, majd a mérések eredményei alapján a formula paramétereinek meghatározása. Ehhez a megfelelő jelfeldolgozási lépéseket a mellékelt *generate_model.m* című MATLAB-szkriptben tettük meg.

2.1 A matematikai modell meghatározása

A bevezetőben tárgyalt matematikai modell megalkotása előtt megvizsgáltuk a rendelkezésünkre álló mérési eredményeket. A fájlok az alábbi zenei hangokat tartalmazzák:

- *rhodes_test_55.mp3*: 55 Hz-es zenei ***a*** hang (kontra ***a***)
- *rhodes_test_110.mp3*: 110 Hz-es zenei ***a*** hang (nagy ***a***)
- *rhodes_test_220.mp3*: 220 Hz-es zenei ***a*** hang (kis ***a***)
- *rhodes_test_440.mp3*: 440 Hz-es zenei ***a*** hang (egyvonalas ***a***)

Beolvastuk, majd eltároltuk a hangmintákat és a 44,1 kHz-es mintavételi frekvenciát. A legkisebb ütéserősségű és frekvenciájú leütés időtartománybeli és spektrális reprezentációját az 2.1. ábra mutatja. Itt megfigyelhetjük, hogy exponenciálisan lecsengő, harmonikus jellel van dolgunk, melynek Fourier-együtthatói, valamint lecsengésének mértéke egyaránt függ a leütéserősségtől és a leütött hang frekvenciájától. Matematikai modellünk megalkotásához ezért az additív szintézistechnika alapfeltételezésével éltünk, miszerint az alapharmonikust és annak egész számú többszöröseinek súlyozott összegét egy exponenciálisan lecsengő taggal szorozva közelíthetjük meg a kívánt hangjelet. Ezzel lényegében elhanyagoltuk a 2.1. ábra spektrumán is látható kismértékű eltérést a felharmonikus arányoktól.



2.1. ábra. *F = 1* ütéserősségű, *f = 55* Hz frekvenciájú mérési eredmény az idő- és frekvenciatartományban

Az alkalmazott additív szintézisstratégia alapfeltételezéseként a gerjesztett test rezgéseinek leírása szolgál. Ennek értelmében egy impulzussal gerjesztett és magára hagyott mechanikai rendszer a rezonanciafrekvenciáin rezeg tovább, majd a veszteségek miatt ez a rezgés exponenciálisan lecseng. Egy húr impulzusválaszaként jellemzően az alapfrekvencia egész számú többszörösei szólalnak meg, egy rúd esetén azonban jellemző az ettől való eltérés, más néven inharmonicitás. A 2.1. ábra spektrumán megfigyelhető, hogy ennek mértéke elhanyagolhatóan kicsi, ezért a jel feltételezett, általános matematikai leírása a következő:

$$s(F,f,t) = \sum_{n=1}^{N} A_n(F,f) \cdot cos(n \cdot 2\pi f \cdot t) \cdot e^{-\tau(F,f)\cdot t} \qquad (2.1)$$

Feltételezésünk szerint tehát egy tetszőleges *F* leütéserősségű és *f* frekvenciájú hangjel meghatározandó paraméterei a lecsengést jellemző $\tau(F, f)$ együttható, valamint az *N* db harmonikus komponensének $A_n(F, f)$ súlyozótényezője.

2.2 Az időbeli burkoló előállítása

Az időbeli burkológörbéről már megállapítottuk, hogy egy olyan lecsengő exponenciális függvény, amely a leütéserősségtől és a frekvenciától egyaránt függ. Célunk tehát a (2.1) egyenletben szereplő exponenciális kifejezés *τ(F, f)* paraméterének meghatározása tetszőleges *F* és *f* értékre:

$$b(F,f) = e^{-\tau(F,f)\cdot t} \tag{2.2}$$

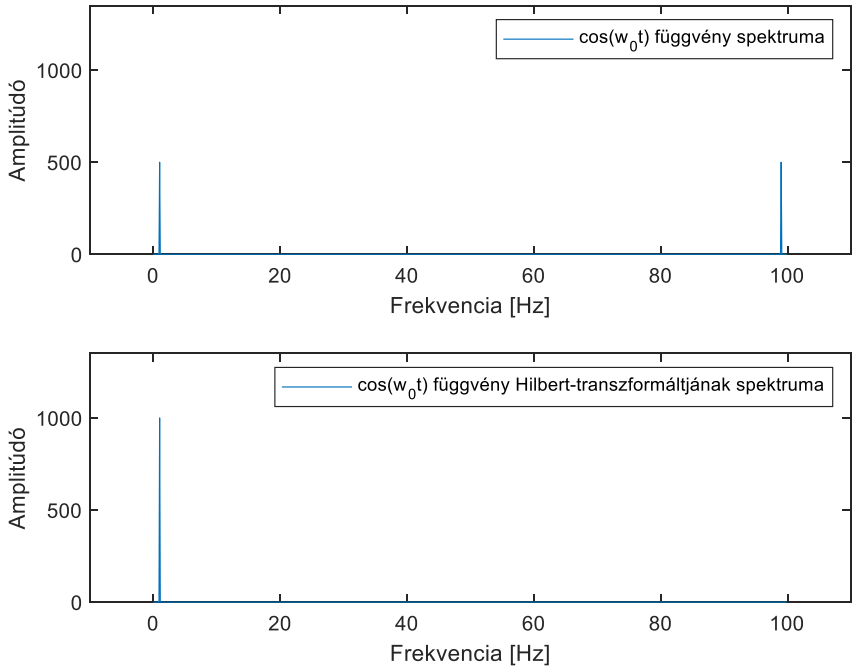
Ehhez először a mérési eredmények burkolóit igyekeztünk minél pontosabban megbecsülni, majd ezekre egy exponenciális görbét illesztve kiértékeltük a τ paraméter értékét minden mért *F* és *f* esetére.

2.2.1 Burkolóbecslés

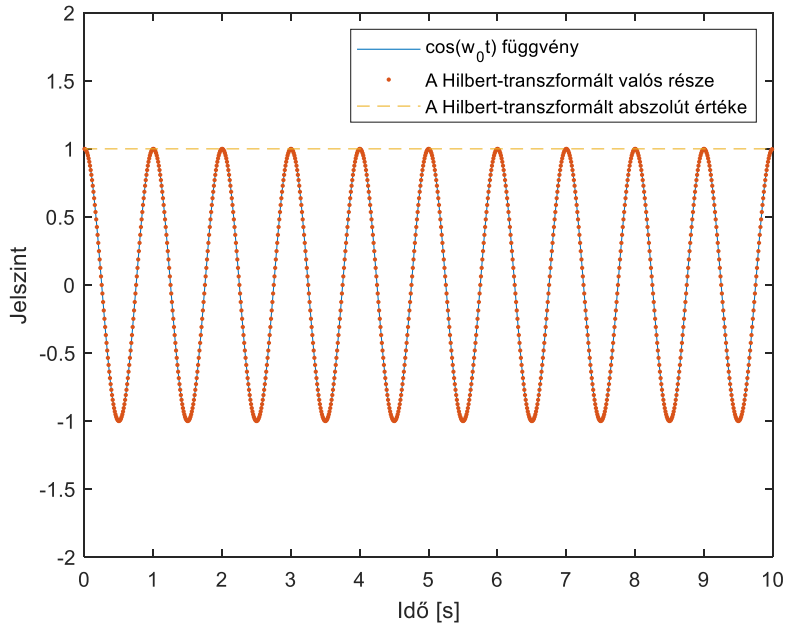
Az időbeli burkológörbe becsléséhez első lépésként a Hilbert-transzformációt alkalmaztuk. Ez a transzformáció egy olyan lineáris művelet, mely az egyoldalas Fourier-transzformáció valós és képzetes részei közti kapcsolatot reprezentálja [10]:

$$\mathcal{H}\{f(t)\} = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{f(\tau)}{t-\tau}d\tau \tag{2.3}$$

Belátható, hogy a transzformált függvény spektruma nem lesz szimmetrikus, ahogyan azt a 2.2. ábra is mutatja. Ennek megfelelően a kapott jel valós része megegyezik az eredeti jellel, abszolút értéke pedig a jel burkolójával, ezt a 2.3. ábra szemlélteti. Az említett példák a transzformáció ezen tulajdonságait mutatják egy 1 Hz frekvenciájú koszinusz-időfüggvény esetére:

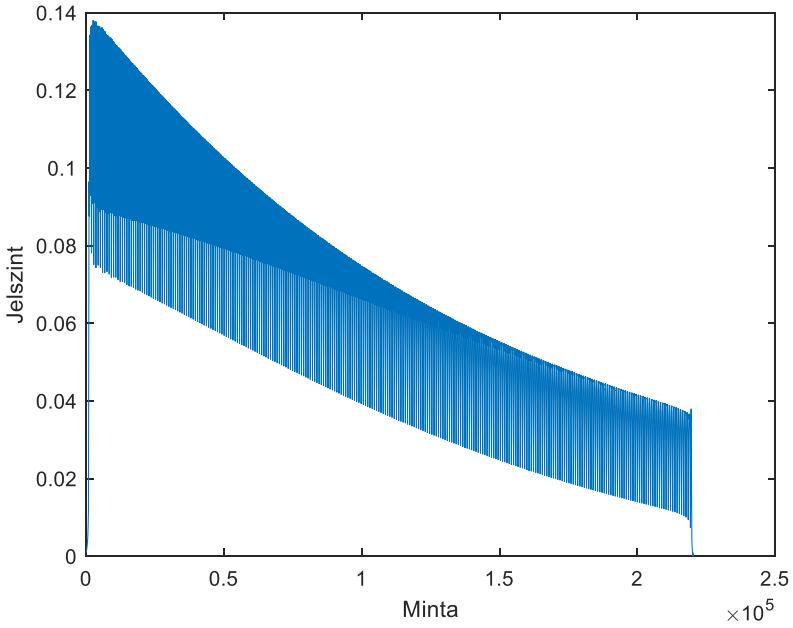


2.2. ábra. Koszinuszfüggvény és Hilbert-transzformáltjának spektruma



2.3. ábra. Koszinuszfüggvény és Hilbert-transzformáltjának valós része, valamint abszolút értéke

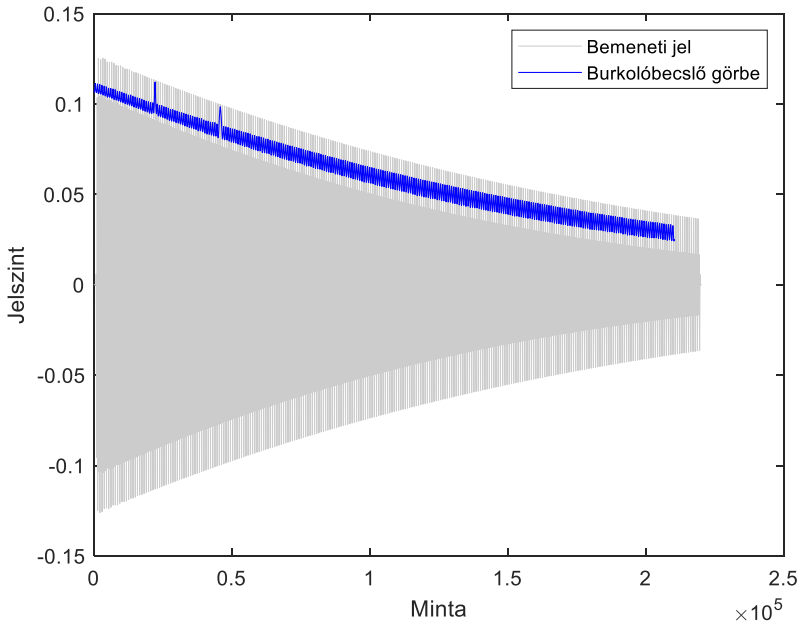
Elsőként képeztük tehát a bemeneti háromdimenziós mátrixunk soraiban tárolt mérések Hilbert-transzformáltjának abszolút értékét. Az így kapott vektor a 2.4. ábra görbéjén látható. A burkoló természetesen még korántsem tökéletes, ezért további műveletek elvégzésével igyekeztünk pontosítani rajta.



2.4. ábra. *F = 1* leütéserősségű és *f = 55 Hz* frekvenciájú mérési eredmény Hilbert-transzformáltjának abszolút értéke

Látható, hogy ha összekötnénk a függvény „felső peremét”, akkor az így kapott görbe már jól közelítené a burkolót. Ehhez először megkerestük a transzformált függvény lokális maximumhelyeit és a hozzájuk tartozó értékeket, majd egy *spline* interpoláció segítségével becsültük meg a köztük lévő, ismeretlen pontok értékeit. Azért ezt a közelítést választottuk, mert így a lineáris interpoláció esetéhez képest jóval simább görbét illeszthettünk a pontokra. A nulladik minta közelében az intervallumon kívül egy extrapolációt is végrehajtottunk, hiszen ez a szakasz nem két interpolációs pont közé esik.

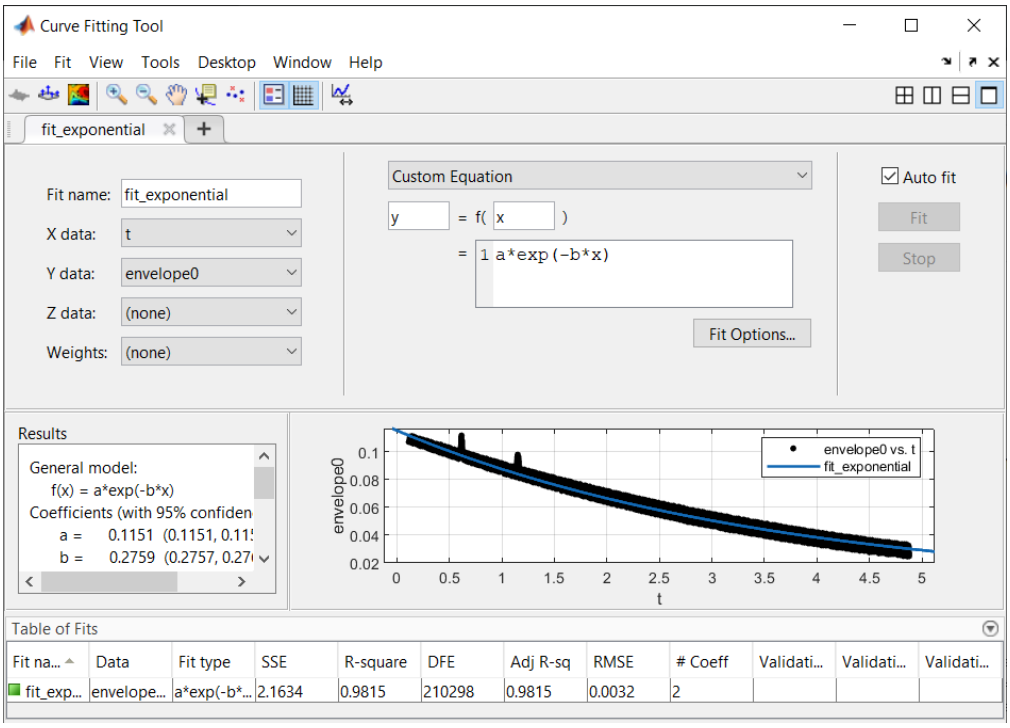
A kapott görbét ezután egy Hann-ablakkal konvolválva elsimítottuk, így valójában egy aluláteresztő szűrést hajtottunk végre. Az így létrehozott új vektorba pedig már csak a kapott eredmény lényegi részét tároltuk el, vagyis néhány mintát eltávolítottunk a jel elejéről és végéről. A 2.5. ábra mutatja, hogy a kapott görbe már jóval pontosabban közelíti az időbeli burkolót.



2.5. ábra. A burkolóbecslő görbe (*F = 1* és *f = 55 Hz* esetén)

2.2.2 A mért burkológörbék közelítése

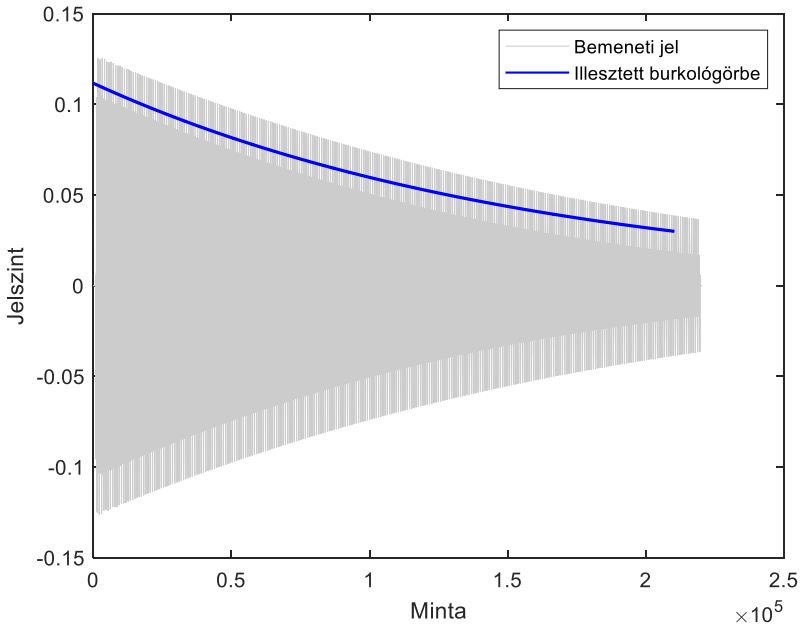
Az így előállított görbéket már megközelíthetjük a kívánt exponenciális függvénnyel. Célunk tehát, hogy a burkolóbecslő görbékre a (2.2) egyenlet formuláját illesszük, majd az exponenciális kitevő együtthatóját rendre eltároljuk a mért burkológörbék τ együtthatóiként. Az illesztést a MATLAB *Curve Fitting Tool* segítségével valósítottuk meg. Az eszköz fő funkciója, hogy a mérési pontokra illesztett görbe paramétereit úgy határozza meg, hogy ezzel minimalizálja az illesztés révén fellépő hibát. Ehhez először kiválasztottuk a szükséges X és Y adatokat, majd a *Custom Equation* mezőibe beírtuk a kívánt formulát (2.6. ábra). Ezután legeneráltuk az illesztést megvalósító MATLAB-függvényt (*fit_exponential.m* a mellékletben).



2.6. ábra. A *Curve Fitting Tool* használata (*fit_exponential*)

A függvényt a kívánt görbére meghívva megkaptuk az illesztés eredményét. Ez az eredmény egy *cfit*[1] típusú objektum, melynek *b* attribútumát elhelyeztük a mérési eredmények τ együtthatóit tároló mátrixban. A paramétereket az illesztés formulájába helyettesítve láthatjuk, hogy jól közelítettünk. A 2.7. ábra mutatja a legkisebb leütéserősségű és frekvenciájú mérési eredményt és az arra illesztett burkológörbét.

[1] A *cfit* (*curve fit*) típusú objektum egy görbeillesztés eredménye, melynek attribútumai a görbét leíró formula paraméterei.



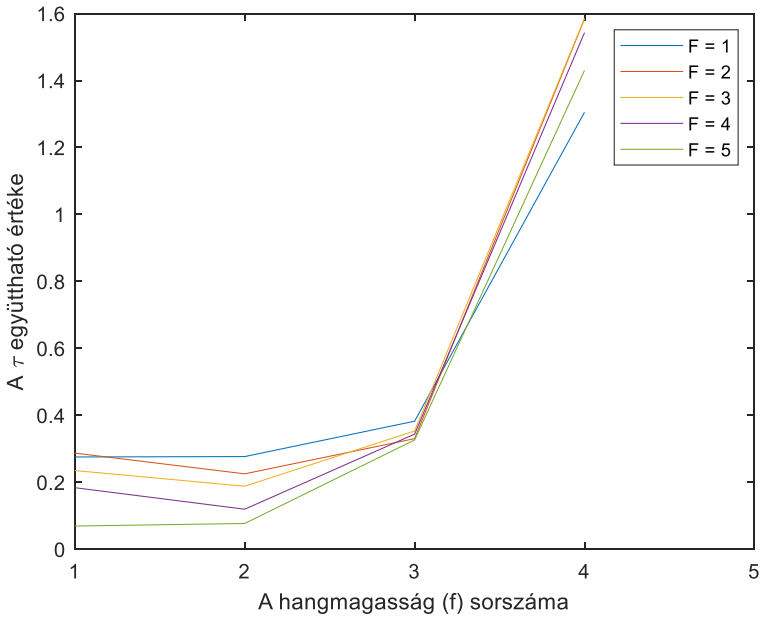
2.7. ábra. Bemeneti jel és illesztett burkológörbéje (*F = 1* és *f = 55 Hz* esetén)

A görbéket ábrázolva azt az érdekességet tapasztaltuk, hogy a 440 Hz-es leütések lecsengése lényegesen gyorsabb, mint a többi leütésé. A jelenség a 2.8. ábra eredményein is jól látható, valamint később a szintetizált hangban is kifejtette a hatását, ugyanis az extrapolált eredmény lecsengése nagyobb frekvenciákon irreálisan gyors volt. Ezért az együtthatómátrix negyedik sorában egy 0,75-tel való szorzással manuálisan csökkentettük le az együtthatók értékét.

2.2.3 A burkolóegyütthatók illesztése

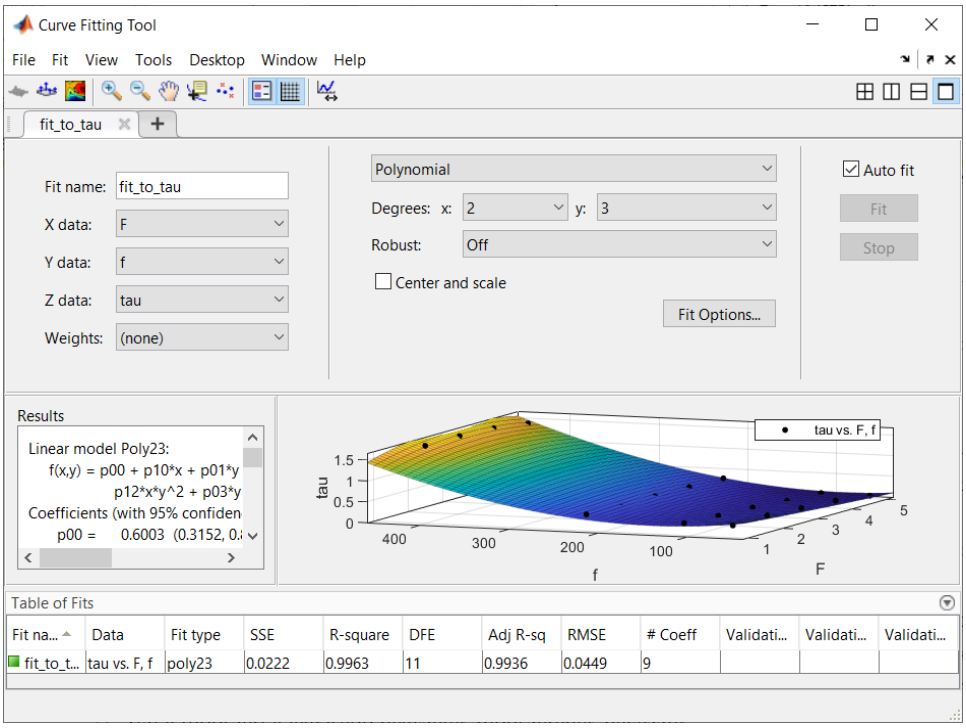
Együtthatómátrixunk tehát 4×5-ös méretű, ugyanis kizárólag az öt mért leütéserősségre és a négy mért hangmagasságra tartalmazza a τ paraméter pontos értékét. Utolsó lépésként célunk, hogy ezt meghatározzuk tetszőleges F és f értékre. Ennek során a mátrixra egy olyan felületet illesztettünk, amely a mérési eredményeinkre közelítőleg a kiszámolt értékeket adja vissza, emellett azonban egy analitikus formulával is leírható F és f függvényében. Ezt a formulát ezúttal egy polinomként határoztuk meg:

$$\tau(F,f) \approx \sum_{\substack{i,j\geq 0 \\ i+j\leq n}} p_{ij} F^i f^j \tag{2.4}$$



2.8. ábra. Különböző ütéserősségű τ értékeink a hangmagasság függvényében ábrázolva

Az illesztést ezúttal is a *Curve Fitting Tool* segítségével tettük meg. Először beírtuk a megfelelő X, Y és Z adatokat, hiszen ezúttal háromdimenziós illesztést hajtottunk végre, majd az illesztés típusaként a *Polynomial* lehetőséget választottuk (2.9. ábra). Ezutáni kísérletezéseink során azt tapasztaltuk, hogy egy *F*-ben másodfokú, *f*-ben harmadfokú polinom már megfelelően közelíti az együtthatóinkat, ezért kitöltöttük a *Degrees* mezőket, majd legeneráltuk az illesztést megvalósító függvényt (*fit_to_tau.m* a mellékletben).



2.9. ábra. A *Curve Fitting Tool* használata (*fit_to_tau*)

A függvényhívás után az illesztés eredménye egy *sfit*[2] típusú objektum, amely azon polinom p_{ij} együtthatóit tárolja, amely legkisebb négyzetes hiba értelemben a legjobban illeszkedik mérési eredményeinkre. Meghatároztuk tehát a célként kitűzött $\tau(F, f)$ paramétert tetszőleges F és f értékre az alábbi formában:

$$\begin{aligned}\tau(F,f) = {} & p_{00} \\ & + p_{10}F + p_{01}f \\ & + p_{20}F^2 + p_{11}Ff + p_{02}f^2 \\ & + p_{21}F^2f + p_{12}Ff^2 + p_{03}f^3\end{aligned} \tag{2.5}$$

2.3 A spektrum vizsgálata és a Fourier-együtthatók előállítása

A jel feltételezett matematikai leírásában szereplő Fourier-együtthatókról megállapítottuk, hogy egyaránt függenek a leütés erősségétől és a megszólaltatott hang frekvenciájától. Célunk ezúttal az, hogy meghatározzuk a (2.1) egyenletben található összeg $A_n(F, f)$ paramétereit:

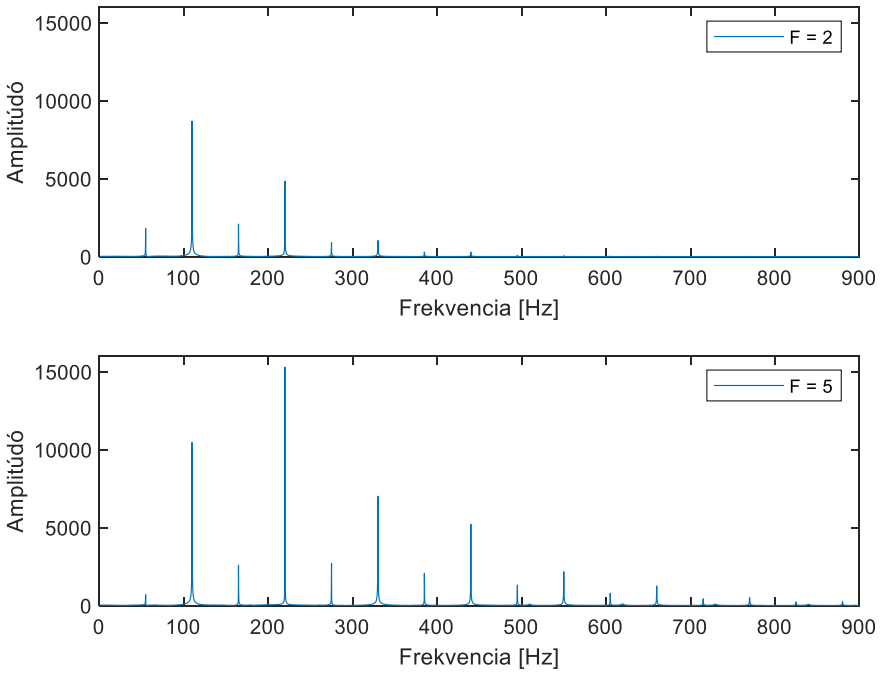
[2] Az *sfit* (*surface fit*) típusú objektum egy felületillesztés eredménye, melynek attribútumai a felületet leíró formula paraméterei.

$$a(F,f) = \sum_{n=1}^{N} A_n(F,f) \cdot cos(n \cdot 2\pi f \cdot t) \tag{2.6}$$

A kifejezésből jól látható az additív szintézis feltételezése, miszerint különböző amplitúdójú és frekvenciájú szinuszhullámok összegeként keletkezik az a rezgés, ami végül az előző fejezetben tárgyalt módon lecseng. Ezen amplitúdók meghatározásához először ábrázoltuk, majd megvizsgáltuk a mért leütések spektrumait. Ezután a Fourier-transzformáció segítségével meghatároztuk őket minden mért F és f értékre, majd a kapott spektrumkomponensekre a megfelelő fokszámú polinomot illesztve általánosíthattuk őket az előző fejezetben is tárgyalt módon, egy analitikus formulaként. A harmonikus frekvenciák értékét az alapfrekvencia egész számú többszöröseiként közelítettük.

2.3.1 A mért spektrum vizsgálata

Mérési eredményeink spektrumkomponenseit úgy tekinthetjük meg, ha ábrázoljuk a bemeneti mátrix valamely sorának gyors Fourier-transzformáltját. A kapott spektrumképet a 2.10. ábra mutatja. Látható, hogy minél nagyobb az ábrázolt leütés erőssége, annál jelentősebb a nagyobb frekvenciájú felharmonikusok energiája.



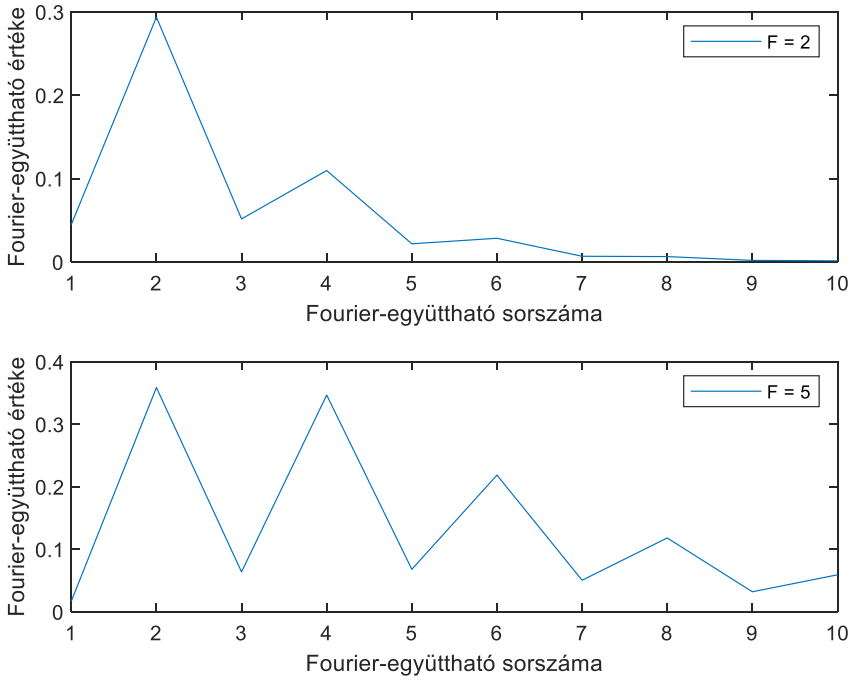
2.10. ábra. *f = 55 Hz* frekvenciájú mérési eredmény spektruma (*F = 2* és *F = 5* esetén)

A (2.1) egyenletben szereplő összeg a Fourier-sor alapfeltételezése miatt van jelen, melynek értelmében tetszőleges periodikus jel előállítható harmonikus jelek összegeként. Mivel a gyakorlatban véges *N* számú komponens is elegendő a spektrum adott pontosságú leírásához, ezért ezt a számot egy 10-re választottuk meg.

A mért amplitúdók meghatározásához a 2.10. ábra csúcsainak nagyságára lenne szükségünk. Triviális, azonban meglehetősen körülményes megoldás lenne ezeket egyenként kikeresni, majd leolvasni minden egyes leütéserősségre és frekvenciára, ehelyett inkább a jel Fourier-sorát számoltuk ki. Ezzel a módszerrel ugyanis kizárólag a számunkra fontos frekvenciákon értékelhettük ki a jel FFT-csúcsainak nagyságát. Mivel ezeket a frekvenciákat az alapfrekvencia és egész számú többszöröseiként választottuk meg, ezért előállítottuk minden mért hangmagasság transzformációs mátrixát és elvégeztük a transzformációt minden mért leütéserősségre az alábbi módon:

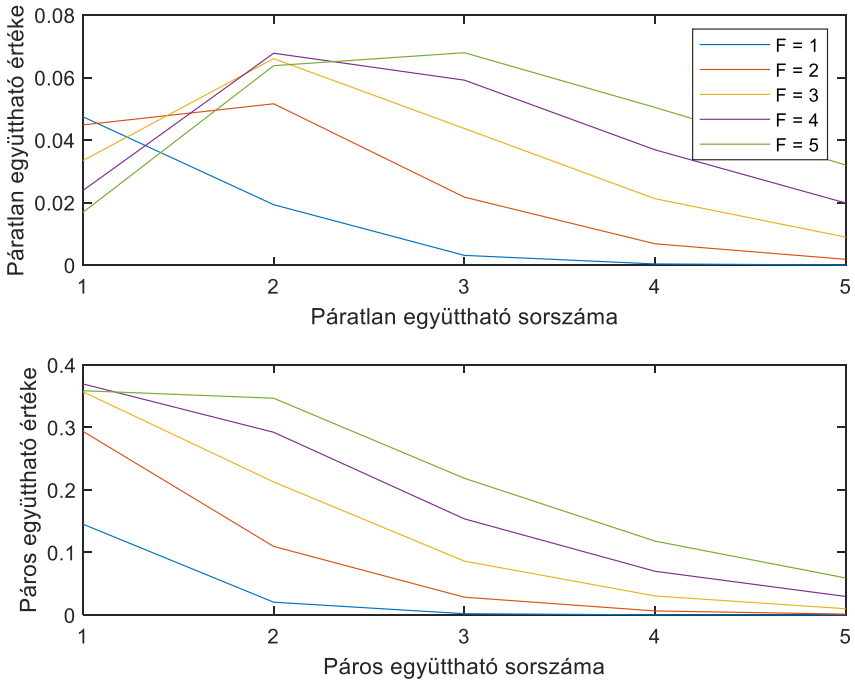
$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} cos(\omega_0 t) \\ cos(2 \cdot \omega_0 t) \\ \vdots \\ cos(N \cdot \omega_0 t) \end{bmatrix} \cdot \begin{bmatrix} \mathrm{i} \\ \mathrm{n} \\ \mathrm{p} \\ \mathrm{u} \\ \mathrm{t} \end{bmatrix} \tag{2.7}$$

Az így létrejött 10×5×4-es mátrix tartalmazza mind a 10 harmonikus együtthatót minden mért *F* és *f* értékre. Ezeket a 2.11. ábra szemlélteti.



2.11. ábra. *f = 55 Hz* frekvenciájú mérési eredmény Fourier-sora (*n = 10*, F = *2* és *F = 5* esetén)

Az ábrát vizsgálva azt tapasztaljuk, hogy a szomszédos komponensek között meglehetősen nagy az amplitúdó-különbség, ilyen gyors változásokat pedig nem igazán lehet alacsony fokszámú polinommal közelíteni. Felismerhetjük azonban, hogy ez külön a páratlan és a páros komponensekre sokkal kevésbé jellemző, így lehetőségünk nyílik arra, hogy külön-külön vizsgálva megközelítsük őket egy-egy simább felülettel. A jelenség a 2.12. ábra különböző ütéserősségekhez tartozó görbéin is megfigyelhető. A továbbiakban ezért külön kezeljük a páratlan és páros együtthatókat.



2.12. ábra. Páratlan és páros spektrumegyütthatók (*f* = *55 Hz* esetén)

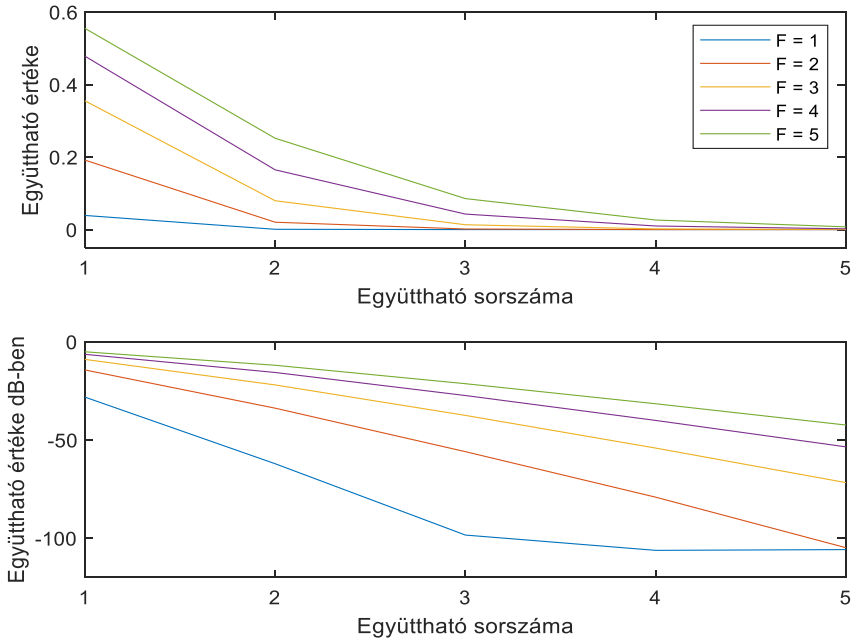
2.3.2 A spektrális együtthatók illesztése

A 2.2. fejezetben láthattuk, hogy a MATLAB *cftool* eszköze segítségével hatékonyan és egyszerűen tudunk különböző illesztéseket megvalósítani. Ennek során a lecsengés együtthatóit tároló 5×4-es mátrixot kellett minden *F* és *f* értékre illesztenünk, ezért egy kétváltozós polinomként leírható felületet határoztunk meg. Ezúttal nehezebb dolgunk van: két 5×5×4-es 3D-mátrixot szeretnénk minden *n*, valamint *F* és *f* értékre illeszteni. Ez azt jelenti, hogy egy háromdimenziós polinommal kéne közelítenünk a kapott spektrális együtthatókat, ami sajnos már túlmutat a *cftool* lehetőségein, hiszen az eszköz maximum kétdimenziós függvényre tud illeszteni.

A probléma áthidalható, ha továbbra is kétváltozós polinomokat illesztünk, ám ezúttal frekvenciánként. Az illesztés eredményeként kapott koefficienseket később, a szintézis megvalósítása során lineárisan fogjuk interpolálni, ezzel előállítva tetszőleges hangmagasságú leütés Fourier-együtthatóit.

Első lépésként tehát a két 3D-mátrixnak mind a négy mért frekvenciához tartozó lapjait elmentettük. Az így létrejövő 5×5-ös mátrixok már adott alapfrekvencia mellett tartalmazzák a páratlan, illetve a páros spektrális együtthatókat minden leütéserősségre, ezekre a mátrixokra pedig már elvégezhettük a kívánt polinomillesztést.

A *cftool* segítségével ezúttal is kísérletezéssel igyekeztünk meghatározni az *n* és *F* paraméterek fokszámát. Ennek során azonban azt tapasztaltuk, hogy a közelítés minden egyes próbálkozás esetén jelentős hibákat eredményezett, ezért valahogyan egyszerűsítenünk kellett ezt a felületet. Mivel a spektrális együtthatókban valamilyen lecsengő viselkedés fedezhető fel, ezért kézenfekvő volt értéküket decibelbe átváltani, ugyanis így ez a felület sokkal inkább egy sík felülethez közelít (2.13. ábra).



2.13. ábra. Páros spektrumegyütthatók és decibelbe átszámított értékük (*f = 220 Hz* esetén)

Látható, hogy a decibelbe való átváltásnak köszönhetően egy lényegesen simább felülettel, vagyis alacsonyabb rendű polinommal is meg fogjuk tudni közelíteni a Fourier-együtthatókat. Ezt a feltevést végül kísérletezéseink is igazolták: a *cftool* eszköz segítségével *F* szerint másodfokú, *n* szerint pedig ötödfokú polinom illesztésével jó közelítést adhattunk az együtthatókra:

$$\begin{aligned} A_n(F,n) = {} & p_{00} \\ & + p_{10}F + p_{01}n \\ & + p_{20}F^2 + p_{11}Fn + p_{02}n^2 \\ & + p_{21}F^2n + p_{12}Fn^2 + p_{03}n^3 \\ & + p_{22}F^2n^2 + p_{13}Fn^3 + p_{04}n^4 \\ & + p_{23}F^2n^3 + p_{14}Fn^4 + p_{05}n^5 \end{aligned} \tag{2.8}$$

Az így legenerált függvény (*fit_to_spectrum.m* a mellékletben) visszatérési értéke tehát egy-egy *sfit* objektum minden mért frekvenciára. Ezzel két cella tömböt hoztunk létre, melyek egyaránt négy-négy cellával rendelkeznek: minden cella egy adott frekvenciához tartozó *sfit* objektumot tartalmaz. Ahhoz, hogy az ezekben lévő p_{ij} együtthatókat végül tetszőleges frekvenciára is meghatározhassuk, a szintézis során a már említett lineáris interpolációt fogjuk alkalmazni.

2.4 A modell létrehozása

Meghatároztuk a felállítani kívánt modell minden számunkra lényeges tulajdonságát. Az előállított paramétereket egy MATLAB-struktúra attribútumaiként (mezőiként) tároltuk el. Ezekre a mezőkre névvel hivatkoztunk, értéküknek pedig a kívánt változók értékékét adtuk.

ADAT	MEZŐ NEVE	MEZŐ TARTALMA	MEZŐ TÍPUSA
A mért frekvencia	*f_measured*	*f*	4×1 *double* vektor
A mért leütéserősség	*F_measured*	*F*	1×5 *double* vektor
A $\tau(F, f)$ paraméter polinomegyütthatói	*fit_tau*	*tau_coefs*	*sfit*
Az $A_n(F, f)$ paraméter páratlan spektrális polinomegyütthatói	*fit_odd_coefs*	*coef_odd*	1×4 *sfit*-cella tömb
Az $A_n(F, f)$ paraméter páros spektrális polinomegyütthatói	*fit_even_coefs*	*coef_even*	1×4 *sfit*-cella tömb

2.1. táblázat. A modell struktúra felépítése

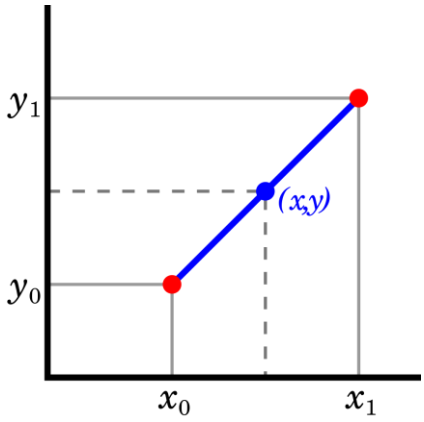
3 A szintézis megvalósítása MATLAB környezetben

Eddigi fejezeteinkben azt tárgyaltuk, hogy mérési eredményeinket hogyan tudjuk tetszőleges billentyűleütésre általánosítani. A melléklet *generate_model.m* című szkriptjében megvalósított jelfeldolgozási lépések eredményeképpen előállítottuk a jel feltételezett matematikai alakjának paramétereit, melyeket a 2.1. táblázatban tárgyalt struktúra mezőiben tároltuk el. Jelen fejezetben kiértékeljük a formulát tetszőleges, előre meghatározott F_0 leütéserősségre és f_0 frekvenciára, majd a szintetizált hangot meghallgatva eldönthetjük, hogy modellünk később megállja-e a helyét VST-pluginként. A szintézis lépéseit a mellékelt *resynthesis.m* című MATLAB-szkriptben tettük meg.

3.1 A spektrális polinomegyütthatók interpolációja

Első lépésként betöltöttük az elkészített modellt, ugyanis a hang előállításához kizárólag a struktúrában tárolt adatokra van szükségünk. Ezután két mátrixot hoztunk létre, amelyekbe külön elhelyeztük a páratlan és páros spektrális polinomegyütthatókat. Ezeknek a mátrixoknak négy sora van, ugyanis a polinomillesztés során csak a négy mért frekvenciára értékeltük ki a koefficienseket.

Ahhoz, hogy tetszőleges magasságú hang együtthatóit meghatározhassuk, az előző fejezet végén már említett lineáris interpoláció eszközét alkalmaztuk, vagyis két mért frekvencia között úgy közelítettük a polinomegyütthatókat, hogy azokra egy egyenest illesztünk. Az interpoláció formulája könnyen levezethető a 3.1. ábra alapján.



3.1. ábra. A lineáris interpoláció szemléltetése [11]

Adott az ábrán is látható két mérési pont, (x_0, y_0) és (x_1, y_1). Az (x_0, x_1) intervallumon belül lineáris interpolációval, az intervallumon kívül pedig lineáris extrapolációval határozhatjuk meg egy tetszőleges x ponthoz tartozó y értéket. Mivel ez a mérési pontokat összekötő egyenesen helyezkedik el, értéke kiszámítható az alábbi formula segítségével [11]:

$$y = y_0\left(1 - \frac{x - x_0}{x_1 - x_0}\right) + y_1\left(\frac{x - x_0}{x_1 - x_0}\right) \tag{3.1}$$

Az interpolációt megvalósító MATLAB-függvény paramétereként tehát megadtuk a mért frekvenciák vektorát és a hozzájuk tartozó páratlan, illetve páros koefficiensek mátrixát, majd a kívánt f_0 frekvenciát. Az így kapott vektorok már tartalmazzák egy tetszőleges F_0 leütéserősségű és f_0 magasságú hang páros és páratlan spektrális együtthatóit előállító polinom p_{ij} koefficienseit. Ahhoz, hogy ebből végül spektrumkomponenseket értékelhessünk ki, készítettünk belőlük egy végleges *sfit* objektumot. Ennek során először a polinomegyütthatók neveit azonosító p_{ij} sztringeket a mentettük el, majd a vektorok értékeit a megfelelő helyre tároltuk el a végleges, kimeneti *sfit* objektumainkban. Az így kapott objektum mezőiből a (2.8) egyenletnek megfelelően állíthatók elő a kívánt páratlan és páros spektrumegyütthatók értékei decibelben.

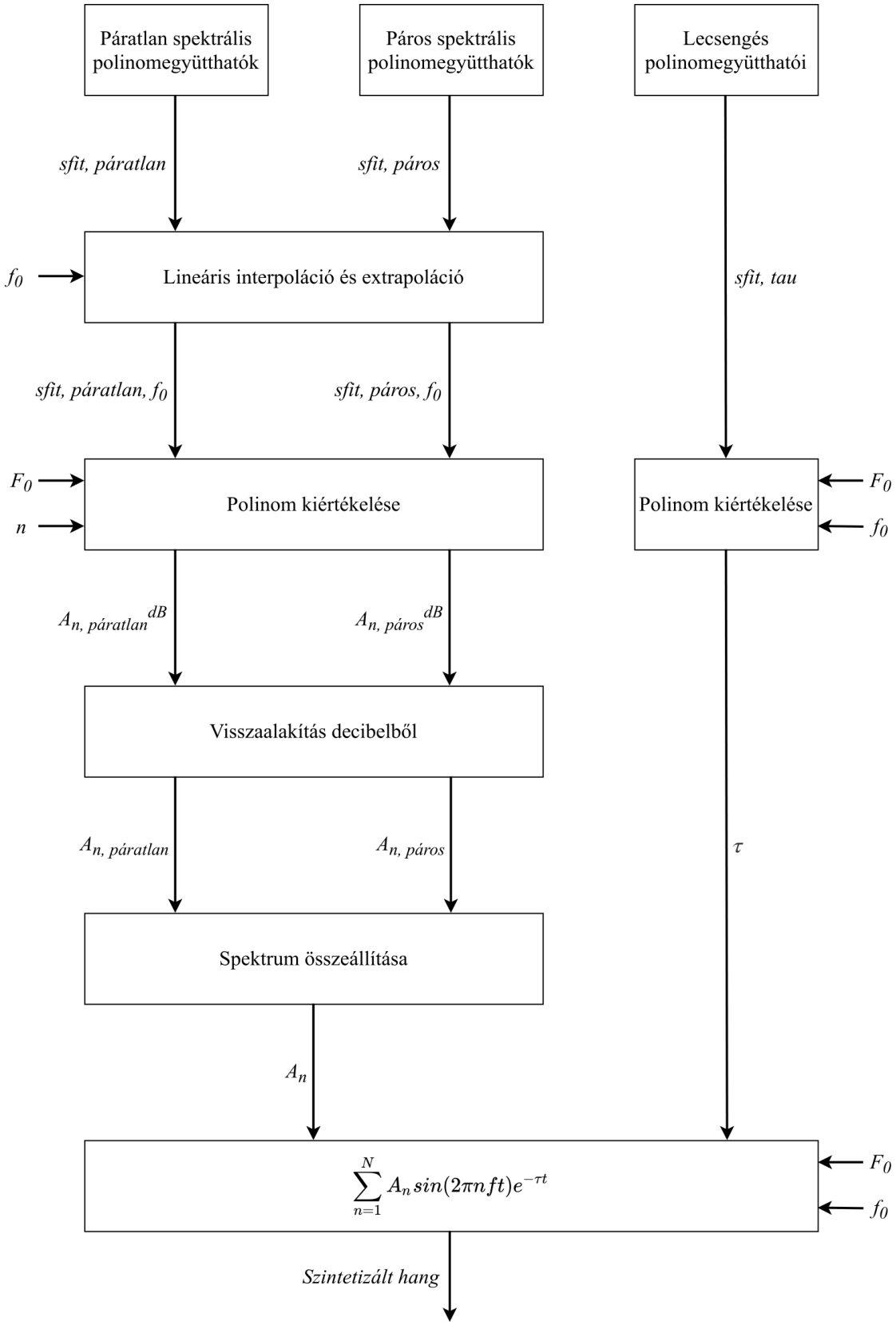
3.2 A szintetizált hang előállítása

A végleges hangjel előállításához közeledve meghatároztuk a Fourier-együtthatók számát, amit ezúttal is 10-re állítottunk be. Ezután már kiértékelhettük a spektrum páratlan és páros együtthatóit a kívánt F_0 leütéserősségben. Az így kapott két vektort „összefésülve” összeállítottuk a teljes spektrumot. Mivel ennek értékei még decibelben vannak, ezért visszaalakítottuk őket.

A (2.1) egyenletben ismertetett matematikai formula kiértékeléséhez ezúttal minden adat rendelkezésünkre áll. Utolsó lépésként létrehoztuk az inverz transzformációhoz szükséges ω_0 alapkörfrekvenciát és az ennek megfelelő koszinusz-időfüggvényt, majd előállítottuk a kívánt leütéserősségű és frekvenciájú kimeneti jelet.

A kapott hangjelet meghallgatva azt tapasztaltuk, hogy a leütés kezdetén és végén egy „pattanás” hallatszik. Ennek oka, hogy a hang felfutása nem a csend szintjéről, lecsengése pedig nem a csend szintjére történik. A jelenségek elsimításaként egy olyan ablakfüggvénnyel szoroztuk meg a jelet, melyet két különböző méretű Hann-ablakból állítottunk össze. Ez tulajdonképpen egy felkeverés (*fade-in*) és egy lekeverés (*fade-out*) művelet megvalósításának is tekinthető, esetünkben a *fade-in* görbéje meredekebb, mint a *fade-out* görbéje.

Az így előállított hangjel már megfelelően közelíti mérési eredményeinket. A 3.2. ábra áttekinti a hangszintézis leglényegesebb lépéseit.



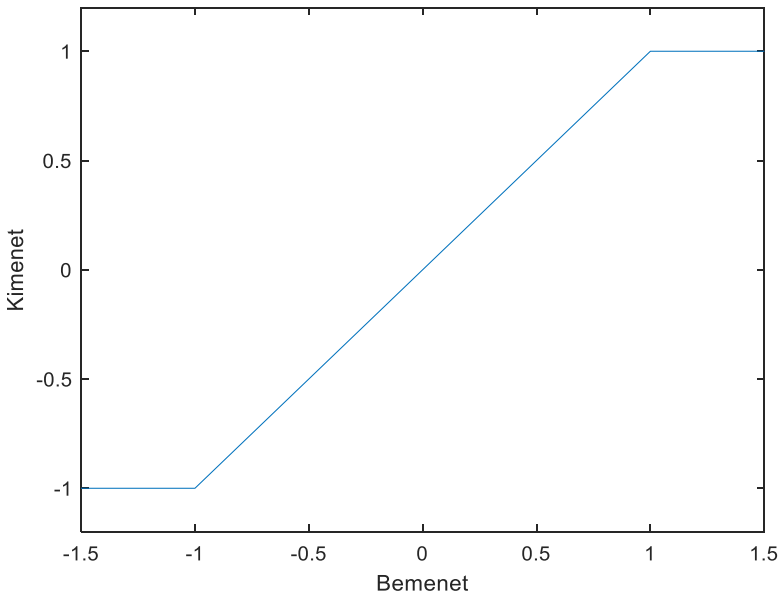
3.2. ábra. A hangszintézis folyamatának összefoglalása

3.3 Soft clipping

Utolsó lépésként egy, a bevezetőben már említett tulajdonságát emeljük ki a hangszernek. Az 1.2 fejezetben kitértünk arra, hogy a megütött hangvilla rezgését a megfelelő hangszedők alakítják elektromos jellé, majd ez végül egy túlvezérelt erősítőn keresztül kerül a hangszóróra. Az erősítő jelenlétét a szintetizált hangban úgy modelleztük, hogy az előállított jelet a melléklet *soft_clipper.m* függvénye segítségével torzítottuk. A függvény alapjául a nemlineáris torzítás (nemzetközi nevén: hard clipping) definíció szerinti formulája szolgál, miszerint egy *x* bemeneti minta *f(x)* nemlineárisan torzított kimenete ±1 között nem változik, -1-nél kisebb bemenetre konstans -1 kimenetet, 1-nél nagyobb bemenetre pedig konstans 1 kimenetet produkál. A torzítás tehát korlátozza a jelet, amikor az túllépi a megengedett értéket:

$$f(x) = \begin{cases} -1, & x \leq -1 \\ x, & -1 \leq x \leq 1 \\ 1, & x \geq 1 \end{cases} \tag{3.2}$$

A hard clipping statikus karakterisztikáját a 3.3. ábra szemlélteti.

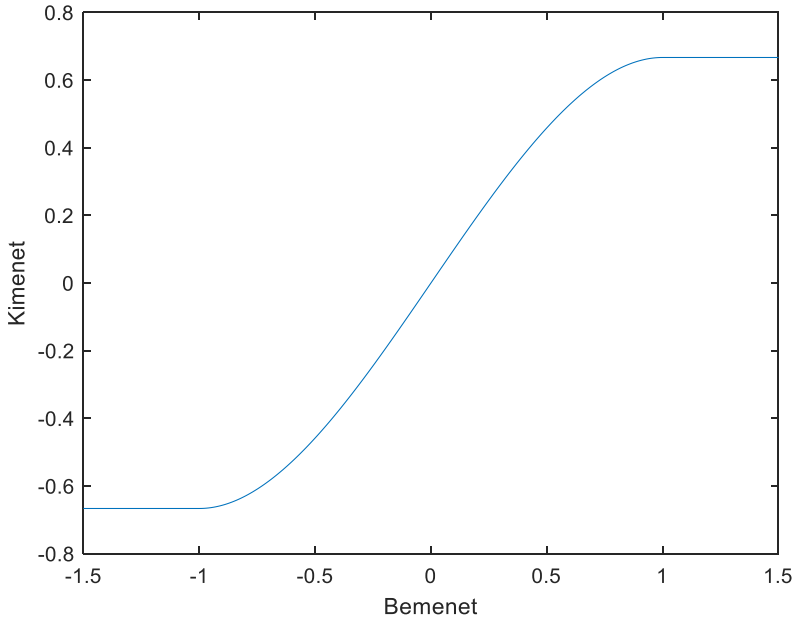


3.3. ábra. A hard clipping statikus karakterisztikája

A soft clipping alapvetően abban tér el a hard clippingtől, hogy karakterisztikája lényegesen simább görbét mutat. Ezért általánosságban a torzítás kifejezésében szereplő köbös nemlinearitás felel [12]:

$$f(x) = \begin{cases} -\frac{2}{3}, & x \leq -1 \\ x - \frac{x^3}{3}, & -1 \leq x \leq 1 \\ \frac{2}{3}, & x \geq 1 \end{cases} \tag{3.3}$$

A soft clipping statikus karakterisztikáját a 3.4. ábra mutatja.

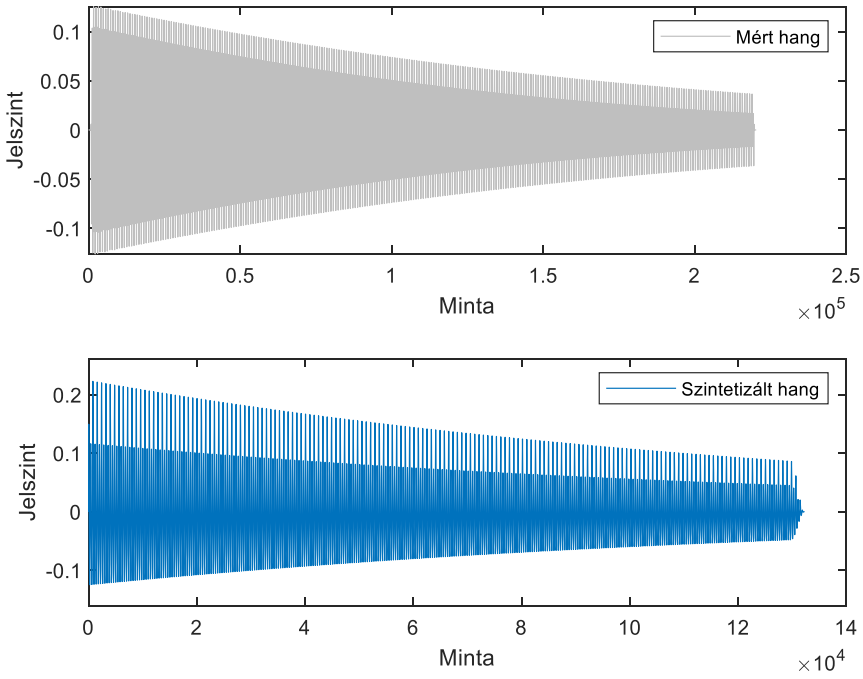


3.4. ábra. A soft clipping statikus karakterisztikája

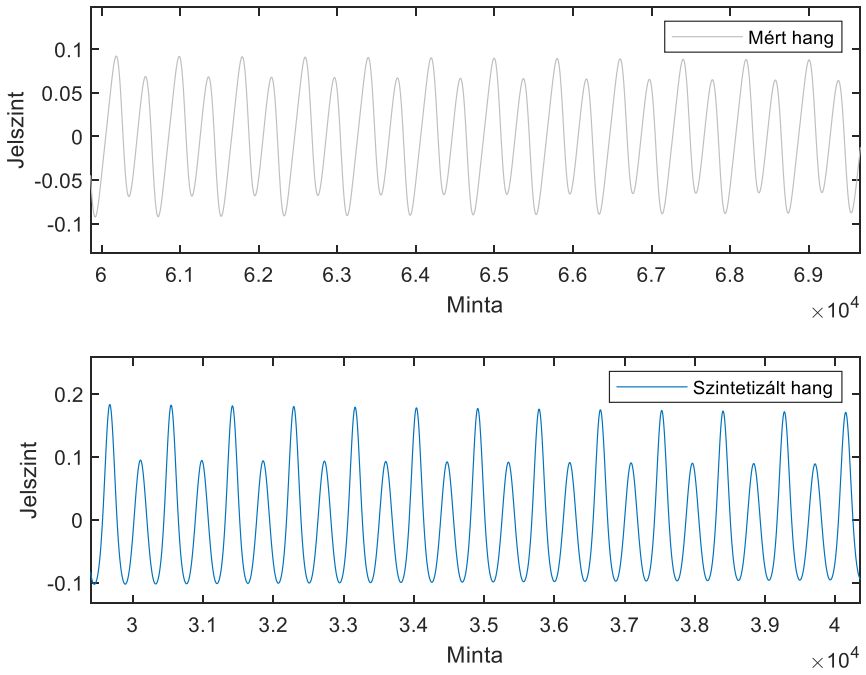
A Fender Rhodes elektromos zongora megszokott torzítását a soft clipper lényegesen pontosabban modellezi, ezért jelen dolgozatban ezt alkalmaztuk. A torzítással kapott hangjel sokkal élethűbben és autentikusabban közelíti a hangszer hangját.

3.4 A mért és a szintetizált hang összehasonlítása

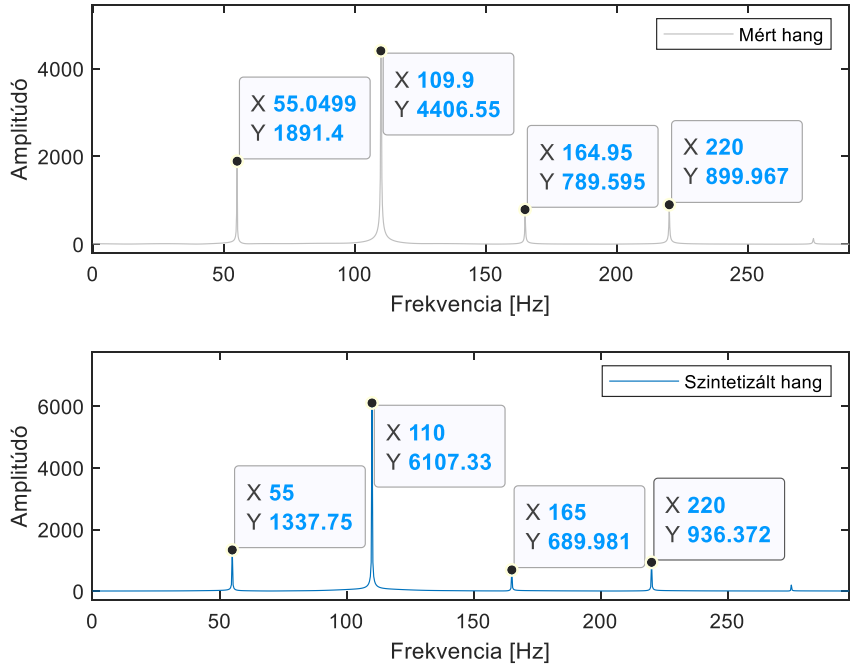
Az alábbi ábrák tanulmányozásával jól láthatók a mért és a szintetizált zongorahang közti különbségek. A könnyen észrevehető eltérések ellenére a két hang meghallgatásának szubjektív élménye nagyon hasonló. Mivel jelen dolgozat célja ezen szubjektív hangélmény reprezentációja, ezért a szintetizált hang alkalmas arra, hogy virtuális hangszerként is megvalósítsuk.



3.5. ábra. A mért és a szintetizált zongorahang összehasonlítása



3.6. ábra. A mért és a szintetizált zongorahang összehasonlítása (közelített ábra)



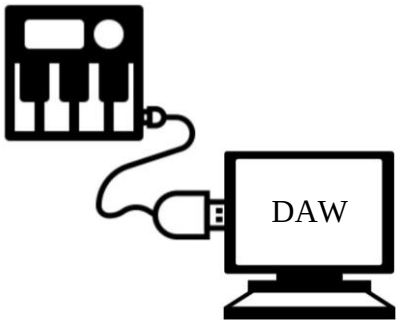
3.7. ábra. A mért és a szintetizált zongorahang spektrumának összehasonlítása

4 VST implementáció

A szintézis során azt tárgyaltuk, hogyan tudjuk a jel feltételezett matematikai alakjának paramétereiből kiindulva előállítani a kívánt billentyűhangot. Az alkalmazott additív szintézistechnika lehetővé teszi egy valós idejű VST plugin implementációját. Célunk ennek megfelelően egy olyan MIDI-vezérelhető, polifonikus szintetizátor-szoftver megvalósítása, amely egy külső billentyűzet vagy kontroller MIDI-üzeneteivel kommunikálva online hangszintézisre képes.

4.1 A szoftver használata

Ahhoz, hogy szoftverünket valós idejű VST-hangszerként használhassuk, a bevezetőben is említett programokra és berendezésekre, valamint azok megfelelő konfigurációjára lesz szükségünk (4.1. ábra).



4.1. ábra. A VST plugin használata

4.1.1 DAW szoftver

Számítógépünkön egy tetszőlegesen választott DAW szoftvert kell futtatnunk, hogy hosztként kezelhessük a kész VST modult. Ezek felhasználói felülete gyakorlatilag egy digitális keverőpult kialakításához hasonlít, így a hangsávokra a megfelelő plugint illeszthetjük, majd felvehetjük a kívánt hanganyagokat. Általánosságban elmondható tehát, hogy a VST eszközök bővítik a DAW alkalmazások funkcionalitását, ezért legtöbbjük képes valós idejű MIDI-üzenetek fogadására, feldolgozására és paramétereik hangolására is.

A dolgozat céljaként kitűzött VST plugint az egyik legnépszerűbb DAW szoftverben, az Ableton Live 10 Suite-ben teszteltem (4.2. ábra).

4.2. ábra. Az Ableton Live 10 Suite felhasználói felülete

4.1.2 MIDI-billentyűzet

A MIDI-billentyűzetek és -kontrollerek olyan berendezések, amelyek MIDI-üzenetek küldésére képesek. Általában USB-, vagy MIDI-kábel segítségével csatlakoztathatjuk őket a használni kívánt PC-hez, vagy külső hangkártyához. Jellemzően zongorabillentyűzettel, ám gyakran nyomógombokkal, potenciométerekkel és tolópotenciométerekkel is felszerelik őket annak érdekében, hogy a billentyűk lenyomására küldött MIDI-üzenetek paraméterei a DAW szoftveren keresztül közvetlenül, hardveresen is vezérelhetők legyenek [7].

A szakdolgozatban készülő szintetizátort Arturia KeyLab 88 billentyűzettel teszteltem (4.3. ábra).

4.3. ábra. Arturia KeyLab 88 MIDI-billentyűzet

4.1.3 PC és hangkártya

MIDI-szoftverszintetizátor használatához természetesen elengedhetetlen egy megfelelő számítógép jelenléte. Jellemző azonban, hogy egy átlagos PC integrált hangkártyája jelentős késleltetést produkál a jelfeldolgozás során. A jelenség kiküszöbölésére született meg az Audio Stream Input/Output (röviden: ASIO) szabvány, melynek segítségével a választott VST hangszer valós időben történő használatakor minimalizálhatjuk a késést. Egyes külső hangkártyák szoftverei tartalmazzák a szükséges ASIO-drivert, ezért stúdiókörülmények között használatuk nélkülözhetetlen.

Jelen szintetizátor-szoftver tesztelése során a Steinberg által kifejlesztett *ASIO4ALL* univerzális driver protokollt használtam.

4.2 A VST fejlesztőkörnyezet

A Steinberg által megalkotott szabvány lehetővé teszi bármely harmadik féltől származó fejlesztő számára, hogy VST plugint készítsen tetszőleges VST hoszt alkalmazáson keresztüli használatra. A piacon elérhető pluginok többnyire Windows platformhoz érhetők el, az Apple rendszerein jellemzően az ún. Audio Unit (röviden: AU) technológia használatos, amely az OS X operációs rendszer részét képezi. Más platformokat (pl.: Linux) egyelőre kevés fejlesztő vett célba [8].

A legújabb VST 3 SDK Steinberg weboldaláról ingyenesen letölthető. A C++ osztályokat tartalmazó csomag egy mögöttes C API-n alapul.

A dolgozat céljaként kitűzött Rhodes-szintetizátort a VST SDK 3.7-es verziójának segítségével valósítottam meg Microsoft Visual Studio fejlesztőkörnyezetben. A fejlesztőkészlet jól dokumentált, valamint számos mintaprojektet tartalmaz, melyek tanulmányozásával megérthető és elsajátítható a VST programozás minden lépése. A csomag CMake fájlokat tartalmaz, melyek futtatásával könnyen összeállítható bármely példaprogram. Az összeállításért tehát a CMake honlapjáról ingyenesen letölthető *cmake-gui* szoftver felel, így a kész build már megnyitható a kívánt fejlesztőkörnyezetben [13].

4.3 A szoftverszintetizátor megvalósítása

4.3.1 A *Note Expression Synth* megismerése

Az implementáció első lépéseként fejlesztőkészlet számos mintaprogramja közül a *Note Expression Synth* névre keresztelt VSTi plugint tanulmányoztuk (4.4. ábra).

A hangszer lényegében egy érintésérzékeny, polifonikus szintetizátor. Könnyen kezelhető és szerkeszthető felhasználói felülete a Steinberg VSTGUI eszközkészletének felhasználásával készült. Speciális funkciója, hogy támogatja a *note expression* technológia nyújtotta lehetőségeket. Ez azt jelenti, hogy a megszólaló jel egyes tulajdonságait nem csak a plugin felhasználói felületén látható paraméterek kézi változtatásával módosíthatjuk, hanem a használt MIDI-kontroller egyes vezérlői segítségével is. Ilyen vezérlők többek között a *Modulation* és a *Pitch* kerekek, valamint egyes gombok, potméterek és faderek is. Példaként említhetjük a *Pitch* kerék elforgatását, amelynek hatására a plugin *Tuning* paramétere is azonos mértékben módosul.

4.4. ábra. A *Note Expression Synth* felhasználói felülete

A *Note Expression Synth* képes szinuszjelet, háromszögjelet, négyszögjelet és vörös zajt generálni. Esetünkben csak a szinuszjel érdekes, ugyanis a modellünk illesztett tulajdonságai is egy egyszerű szinuszjelre vonatkoznak. Célunk tehát, hogy ezúttal C++ nyelven implementáljuk a modellalkotás során kiszámolt paramétereket a mintaplugin hangját előállító *NoteExpressionSynth::Voice* osztályba, majd a kimeneti jelet a (2.1) egyenletnek megfelelő formára módosítani.

4.3.2 Lineáris interpoláció és extrapoláció

Első lépésként statikus konstans tömbök formájában deklaráltuk az illesztett modell paramétereit a *VoiceStatics* osztályba. Ezután a szintézis egyik legfontosabb lépése, a spektrális polinomegyütthatók közti lineáris interpoláció és extrapoláció következett, melyet ezúttal manuálisan tettünk meg a 3.1. fejezetben tárgyalt módon. A kívánt hangmagasságot a frekvenciatáblázat *pitch* változójával való címzéssel értük el, majd erre az értékre végeztük el a szükséges utasításokat a *Voice* osztály *NoteOn* függvényében.

4.3.3 A valós idejű hangszintézis megvalósítása

A szintetizátor forráskódját tanulmányozva az tapasztaltuk, hogy a leütéserősség 0 és 1 közti számként van jelen, ezért a spektrum előállításához a *velocity* változó ötszörösével számoltunk. Először kiértékeltük a páratlan, majd a páros spektrális együtthatókat a (2.8) egyenletnek megfelelően, majd visszaállítottuk az értéküket decibelből. A végleges spektrum tömbje már összeállítható az így kapott értékekből. Hasonlóan jártunk el a burkoló kiértékelésekor, természetesen ezúttal a (2.5) egyenletnek megfelelően.

A végleges hangjel előállításához a *Voice* osztály *process()* függvényében szereplő szinuszjelet vettük alapul. Mindössze tehát annyi változtatást tettünk, hogy a szinuszt szoroztuk a spektrumegyütthatókkal, majd az így kapott jelet utolsó lépésként szoroztuk a kívánt exponenciális kifejezéssel.

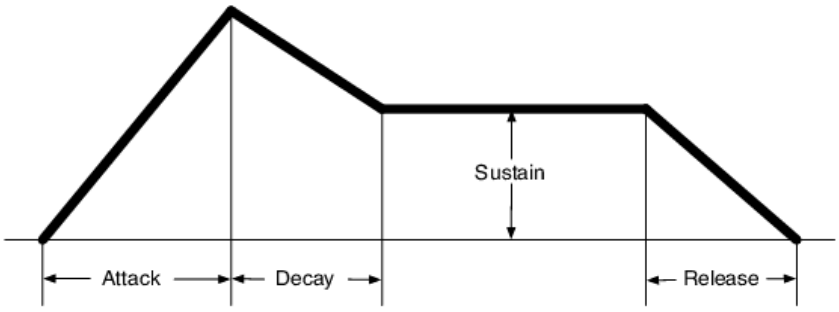
5 Összefoglalás

5.1 Az illesztett modell értékelése

A MATLAB-modell implementációja során azt a célt tűztük ki, hogy minél pontosabban közelítsük mérési eredményeinket. Legfontosabb szempontunk tehát az volt, hogy a szintetizált hangjel a hallgatóság számára kielégítően hasonlítson az eredeti hangszer hangjához, ezért elhanyagoltunk egyes tulajdonságokat, amelyek az emberi fül számára nem hallhatók. Ennek előnye, hogy a számításigényt jelentős mértékben lecsökkenthettük, hiszen a módosítható paraméterek száma tulajdonképpen minimális. Ez természetesen a módszer hátrányaként is tekinthető, ugyanis a piacon elérhető Rhodes-szoftverszintetizátorok felhasználói felülete számos tulajdonság hangolását teszi lehetővé.

5.1.1 A burkoló értékelése

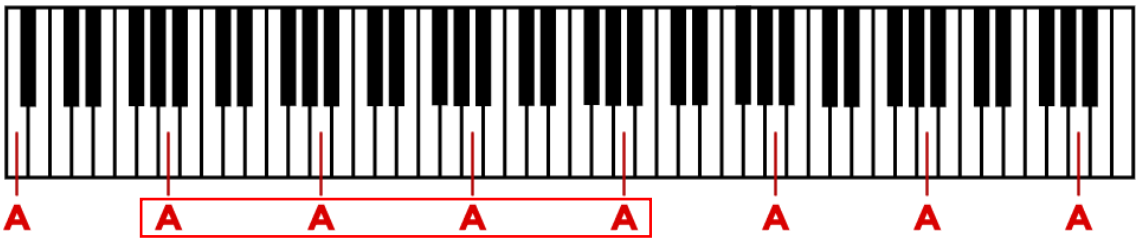
A τ együttható valójában a lecsengést jellemző időállandó reciproka, melyet számításaink és illesztéseink megkönnyítése érdekében nem nevezőként, hanem együtthatóként definiáltunk az exponenciális hatványkitevőben. Ennek meghatározásakor valójában a burkoló négy nagyon fontos tulajdonságát közelítettük: a felfutást, a csillapítást, a kitartást és a lecsengést (nemzetközi nevén: Attack – Decay – Sustain – Release, röviden: ADSR) [5]. A szakaszokat az 5.1. ábra szemlélteti. Ezek a paraméterek a legtöbb szintetizátorban vezérelhetők, esetünkben azonban egy exponenciális görbeként általánosítottuk őket. Ezen görbét egyértelműen definiálja az előállított τ együttható a leütéserősség és a hangmagasság függvényében.



5.1. ábra. Az ADSR-görbe szakaszai [14]

Mivel a burkolót mindössze a tárgyalt exponenciális görbe alkotja, ezért belátható, hogy az előállított ADSR-görbénk tulajdonképpen csak a csillapítás és a kitartás szakaszokkal rendelkezik. Ez indokolttá és szükségessé tette, hogy a szintetizált hangot a két Hann-ablak segítségével felkeverjük és lekeverjük, így lényegében manuálisan valósíthattuk meg a burkoló felfutását és lecsengését.

Nem ez a művelet volt az egyetlen kézi beavatkozásunk a jel burkolójába. Szintén a szintetizált hang meghallgatásakor tapasztaltuk, hogy a magasabb hangok lecsengésének mértéke túl nagy. Ez a jelenség abból fakad, hogy mérési eredményeink csak négy frekvencián álltak rendelkezésünkre (5.2. ábra), ezek a frekvenciák pedig a zongoraklaviatúra legmagasabb hangjához viszonyítva mind legalább egy nagyságrenddel kisebbek. A három- és négyvonalas oktáv hangjainak burkológörbéi így lényegesen nagyobb eltérést mutatnak az eredeti hangszerhez képest. A 440 Hz-es leütés τ paraméterének manuális csökkentése ezt a jelenséget hivatott kompenzálni.



5.2. ábra. A mérési eredmények helye a zongoraklaviatúrán [15]

5.1.2 A spektrum értékelése

A spektrum előállítása során a harmonikusok frekvenciáit kizárólag az alapharmonikus frekvencia egész számú többszöröseiként közelítettük. Nem vettük tehát figyelembe a hangszerre jellemző inharmonicitást, ugyanis ezek elhanyagolható mértékben térnek el a felharmonikus arányoktól (2.1. ábra és 3.7. ábra). Fontos azonban megjegyeznünk, hogy az inharmonikus komponensek jelenléte nagyban hozzájárul a legtöbb zongora és elektromos zongora sajátos hangzásához [5].

További közelítésként tekinthetjük azt, hogy a szintézis során a kívánt alapfrekvenciájú hang együtthatóit kizárólag a négy mért frekvenciához tartozó koefficiensek közti lineáris interpolációval és extrapolációval állítottuk elő.

5.2 Továbbfejlesztési lehetőségek

A szoftverszintetizátor továbbfejlesztéseként számos ötlet felmerülhet. Egyes jelenségek oka, hogy aránylag kevés mérési eredmény állt rendelkezésünkre. Ha például a klaviatúra teljes tartományán oktávonként egy hangot mérnénk ki, akkor a dolgozatban megtett lépésekkel bizonyosan jobb minőségű szoftverhangszert hozhatnánk létre.

További fejlesztési lehetőség rejlik a Rhodes szintetizátorokhoz használt hangszedők tanulmányozásában. Ezek mérése, átvitelük pontos meghatározása és modellezése jelentős mértékben növelné a megszólaló leütés autentikus jellegét.

A hangszedők kimenetéhez csatlakoztatott túlvezérelt erősítő torzítása nem minden esetben hasonlít az alkalmazott soft clipper-hez. A Rhodes felhasználói körében népszerű erősítők tanulmányozásával, modellezésével és implementációjával tehát ugyancsak pontosabb közelítést adhatnánk a hangszer jellegzetes hangjára.

VSTi plugin fejlesztése kapcsán nem feledkezhetünk meg a tényről, hogy minden egyes változtatható paraméter beiktatása lényegesen bővíti a szoftverhangszer funkcionalitását. Mivel a soft clipper alkalmazásával csak egy adott mértékű torzítást valósítottunk meg, ezért jó fejlesztési irány lehet ehelyett egy módosítható paraméterekkel rendelkező torzítás effekt implementációja. Hozzájuk hasonlóan akár egyéb digitális effektek (pl.: zengetés, kórus, vibrato) megvalósítása is nagy mértékben növelné a játékélményt. A funkciókhoz valós idejű vezérlési lehetőséget biztosíthatunk, ha továbbra is kihasználjuk a *note expression* technológia lehetőségeit.

Köszönetnyilvánítás

Köszönettel tartozom konzulensemnek, Dr. Firtha Gergelynek, valamint Dr. Rucz Péternek, amiért számíthattam segítségükre, ötleteikre és tanácsaikra. Hálás vagyok továbbá Dr. Bank Balázs Lajosnak is, hogy az általa oktatott ismeretek segítették munkám haladását. Végül köszönöm családom és barátaim támogatását, érdeklődését.

Irodalomjegyzék

[1] Wikipedia: *Rhodes piano*, https://en.wikipedia.org/wiki/Rhodes_piano (hozzáférés dátuma: 2020. október 24.)

[2] DM Audio: *Fender Rhodes Stage 73 electric piano*, https://dmaudio.co.uk/product/fender-rhodes-stage-73-electric-piano-2/# (hozzáférés dátuma: 2020. október 24.)

[3] The Rhodes Super Site: *Service manual, Chapter one: The Rhodes tone source*, http://www.fenderrhodes.com/org/manual/ch1.html#1-1 (hozzáférés dátuma: 2020. október 24.)

[4] Arturia.com: *Stage-73 V overview*, https://www.arturia.com/products/analog-classics/stage-73-v/overview (hozzáférés dátuma: 2020. október 26.)

[5] Szigetvári A., Horváth B.: *Bevezetés a zenei informatikába*, Typotex Kiadó, 2014

[6] A. Falaize, T. Hélie: *Passive simulation of the nonlinear port-Hamiltonian modeling of a Rhodes Piano*, https://hal.archives-ouvertes.fr/hal-01390534/document (hozzáférés dátuma: 2020. október 25.)

[7] Wikipedia: *MIDI*, https://en.wikipedia.org/wiki/MIDI (hozzáférés dátuma: 2020. november 15.)

[8] Wikipedia: *Virtual Studio Technology*, https://en.wikipedia.org/wiki/Virtual_Studio_Technology (hozzáférés dátuma: 2020. október 25.)

[9] Steinberg.help: *Note Expression*, https://steinberg.help/cubase_pro_artist/v9.5/en/cubase_nuendo/topics/note_expression/note_expression_c.html (hozzáférés dátuma: 2020. december 4.)

[10] Dr. Huba A., Dr. Lipovszky Gy.: *Méréselmélet*, http://old.mogi.bme.hu/TAMOP/mereselmelet/book.html (hozzáférés dátuma: 2020. november 24.)

[11] Wikipedia: *Linear interpolation*, https://en.wikipedia.org/wiki/Linear_interpolation (hozzáférés dátuma: 2020. december 7.)

[12] Julius Orion Smith: *Physical Audio Signal Processing*, https://ccrma.stanford.edu/~jos/pasp/Soft_Clipping.html (hozzáférés dátuma: 2020. december 6.)

[13] Steinberg: *Software Development Kit, Version 3.7.1. Documentation*, https://steinbergmedia.github.io/vst3_doc/ (hozzáférés dátuma: 2020. december 4.)

[14] ResearchGate: *The elements in an ADSR envelope*, https://www.researchgate.net/figure/The-elements-in-an-ADSR-envelope_fig22_270819567 (hozzáférés dátuma: 2020. november 23.)

[15] Seekpng.com: *Learn the Musical Notes of the White Keys Starting*, https://www.seekpng.com/ima/u2q8e6r5i1w7t4w7/ (hozzáférés dátuma: 2020. november 23.)

Függelék

I. A *generate_model.m* MATLAB-szkript tartalma:

```
%% A mert hangmintak beolvasasa
clear
close all
addpath('samples')

[in_55,  fs] = audioread( 'rhodes_test_55.mp3' );
[in_110, fs] = audioread( 'rhodes_test_110.mp3' );
[in_220,  fs] = audioread( 'rhodes_test_220.mp3' );
[in_440, fs] = audioread( 'rhodes_test_440.mp3' );

%% Az bemeneti matrix eloallitasa
F = (1:5);
f = [55;110;220;440];

i0 = 4.67e4;
L = 2.67e5-i0;
input = zeros(L,5,4);

for n = 1 : 5
    input(:, n, 1) = in_55(i0 + (n-1)*L+1: i0 + n*L);
    input(:, n, 2) = in_110(i0 + (n-1)*L+1: i0 + n*L);
    input(:, n, 3) = in_220(i0 + (n-1)*L+1: i0 + n*L);
    input(:, n, 4) = in_440(i0 + (n-1)*L+1: i0 + n*L);
end

clear in_55 in_110 in_220 in_440

%% Az idobeli burkolo eloallitasa
Amp = zeros(size(input,3),size(input,2));
tau = zeros(size(input,3),size(input,2));
L = 1e3;
T0 = 5e3;

for m = 1 : size(input,3)
    for n = 1 : size(input,2)
        envelope = abs(hilbert(squeeze(input(:,n,m))));
        [vals, ixs] = findpeaks(envelope);
        envelope =
interp1(ixs,vals,(1:length(input(:,n,m))),'spline','extrap');
        envelope = conv(envelope,hann(L)/sum(hann(L)),'same');
        envelope0 = envelope(T0+1:end-T0);
        t = (0:length(envelope0)-1)'/fs + T0/fs;

        fit = fit_exponential(t(1:100:end), envelope0(1:100:end));
        coeffs = coeffvalues(fit);
        Amp(m,n) = coeffs(1);
        tau(m,n) = coeffs(2);
    end
end
```

```
tau(4,:) = 0.75*tau(4,:);
tau_coefs = fit_to_tau(F, f, tau);

%% A spektrum vizsgalata es a Fourier-egyutthatok eloallitasa
Ncoefs = 10;
n_odd = (1:2:Ncoefs);
n_even = (2:2:Ncoefs);
t = (0:size(input,1)-1)/fs;
for i_f = 1 : length(f)
    w = (1:Ncoefs)'*2*pi*f(i_f);
    A = exp(-1i*w*t)/fs;
    for i_F = 1 : length(F)
        Spectrum(:,i_F,i_f) = abs(A*squeeze(input(:,i_F,i_f)));
        Spectrum_odd(:,i_F,i_f) = Spectrum(n_odd,i_F,i_f);
        Spectrum_even(:,i_F,i_f) = Spectrum(n_even,i_F,i_f);
    end
end

coef_odd = [];
coef_even = [];
for i_f = 1 : length(f)
    Sp_odd = squeeze(Spectrum_odd(:,:,i_f));
    Sp_even = squeeze(Spectrum_even(:,:,i_f));
    coef_odd{i_f} = fit_to_spectrum(F, n_odd, 20*log10(Sp_odd));
    coef_even{i_f} = fit_to_spectrum(F, n_even, 20*log10(Sp_even));
end

%% A modell letrehozasa
model =
struct('f_measured',f,'F_measured',F,'fit_tau',tau_coefs,'fit_odd_coefs',[
],'fit_even_coefs',[]);
model.fit_odd_coefs = coef_odd;
model.fit_even_coefs = coef_even;
save('Fender_model','model')
```

II. A *resynthesis.m* MATLAB-szkript tartalma:

```
clear
close all
load Fender_model.mat

Odd_cf_mx = coeffvalues(model.fit_odd_coefs{1});
Even_cf_mx = coeffvalues(model.fit_even_coefs{1});
for i = 1 : length(model.fit_odd_coefs)
    Odd_cf_mx(i,:) = coeffvalues(model.fit_odd_coefs{i});
    Even_cf_mx(i,:) = coeffvalues(model.fit_even_coefs{i});
end

%% Tetszoleges frekvenciak es leuteserossegek meghatarozasa
f0_vec = [110, 164.8, 220, 277.2, 370, 220, 329.6, 220]';
F0_vec = [2 1 1 1 2 1 2 1]'*1;
tempo_vec = [1 1 1 1 1 1 1 1]';

f0_vec = repmat(f0_vec,2,1);
```

```
F0_vec = repmat(F0_vec,2,1);
tempo_vec = repmat(tempo_vec,2,1);

%% Szintezis
for j = 1 : length(f0_vec)
    f0 = f0_vec(j);
    F0 = F0_vec(j);
    fs = 48e3;
    Lt = 3;
    t = (0:Lt*fs-1)'/fs;
    Lw = 50;
    win1 = hann(2*Lw);
    win2 = hann(100*Lw);
    win = [win1(1:Lw);ones(length(t)-Lw*51,1);win2(50*Lw+1:end)];

    coef_odd_out = interp1(model.f_measured', Odd_cf_mx, f0, 'linear',
'extrap');
    coef_even_out = interp1(model.f_measured', Even_cf_mx, f0, 'linear',
'extrap');

    pnames = coeffnames(model.fit_odd_coefs{1});
    fit_odd_out = model.fit_odd_coefs{1};
    fit_even_out  = model.fit_even_coefs{1};
    for i = 1 : length(pnames)
        fit_odd_out.(pnames{i}) = coef_odd_out(i);
        fit_even_out.(pnames{i}) = coef_even_out(i);
    end

    Ncoefs = 10;
    Sp_odd_out = fit_odd_out(F0, (1:2:Ncoefs));
    Sp_even_out = fit_even_out(F0, (2:2:Ncoefs));
    Spectrum_out = [Sp_odd_out; Sp_even_out];
    Spectrum_out = 10.^(Spectrum_out(:)/20);

    w0 = (1:Ncoefs)*2*pi*f0;
    iA = cos(t*w0);

    output = (iA*Spectrum_out).*exp(-model.fit_tau(F0,f0)*t).*win;
    output = soft_clipper(output);
    soundsc(output,fs);
    pause(0.5*tempo_vec(j));
end
```