Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Hálózati Rendszerek és Szolgáltatások Tanszék

Fender Rhodes zongora hanganalízise és -szintézise

Készítette:

Novák Bence

Konzulens:

Dr. Rucz Péter

2022. december 8.

HALLGATÓI NYILATKOZAT

Alulírott *Novák Bence*, 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 autentikált 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.

Budapest, 2022. december 8.

Novák Bence
hallgató

Tartalomjegyzék

Kivonat

A Fender Rhodes széles körben használt elektromos zongora. Arra, hogy a hangzása könnyen hozzáférhető legyen bárki számára, kézenfekvő megoldást nyújthat a Rhodes hangját modellező, számítógép segítségével használható, Virtual Studio Technology (VST) keretrendszerben elkészített szoftverhangszer.

Dolgozatom célja, hogy egy valódi hangszer hangmintáin végzett hanganalízis alapján olyan matematikai modellt alkossak a hangszer hangképzéséről, melynek segítségével a VST szoftverhangszerben elvégezhető a valós idejű hangszintézis. Így az eredeti hangszerhez nagyon közeli hangzást nyújtó Rhodes szoftverhangszert készítettem, amelynek a számításigénye viszonylag kicsi.

Az eljárás során hangfelvételeket csináltam egy Rhodes zongoráról. Az így kapott hangmintákat MATLAB környezetben elemeztem harmonikusonként, maximális amplitúdó és az exponenciális lecsengéshez tartozó időállandó tekintetében. A különböző hangmintákból kapott amplitúdó- és időállandóértékekre polinomokat illesztettem, ezzel megkönnyítve a szintézis során alkalmazott interpolációt. A szintézis a VST plugin megvalósításában történik, ezt C++ környezetben implementáltam. A használat során egy adott billentyű leütése esetén, adott erősséggel, a hangmagasság és a leütéserősség alapján történik az interpolálás. Ezzel meghatározhatók a harmonikusokhoz tartozó maximális amplitúdó- és időállandó-értékeket. Ebből harmonikusonként összeállíthatóvá válik a végső hangminta.

A dolgozat eredményeként az eredeti hangszerhez nagyon hasonló hangot előállító VST szoftverhangszert készítettem.

Abstract

The Fender Rhodes piano is a popular electric piano. To make it's sound accessible to more musicians, a software instrument were created using the Virtual Studio Technology (VST) interface.

The aim of my thesis is to create a mathematical model of the Rhodes's sound production based on audio samples from a real instrument. This model is used for the real time sound synthesis in the VST plugin. Using this method, we can achieve a sound very much similar to the original instrument, and using this software instrument is not computationally intensive.

First, I recorded audio samples of a Rhodes piano. I analysed these samples in MATLAB, measuring the maximal amplitude and the time constant of the exponential decay for the harmonics of the sound. To make the interpolation used in the synthesis process easier, I fitted polynomial curves to the amplitude and time constant data obtained from the analysis of multiple audio samples. The synthesis is implemented in the VST plugin, using the C++ programming language. The interpolation relies on the velocity and pitch of the MIDI data obtained from a key is hit on a MIDI keyboard. This method produces the maximal amplitudes and the time constants for the harmonics of the sound. From these harmonics, the final audio sample is compilable.

The result of this thesis is a VST software instrument, producing a very similar sound to the original Fender Rhodes instrument.

1. A Fender Rhodes

1.1. Motiváció

Bár nem játszom billentyűs hangszereken, mindig is lenyűgözött a Rhodes hangja. Akár általam hallgatott hangfelvételeken, akár amikor a hangszer hangját imitáló szintetizátoron játszott egy zenésztársam, a szívembe lopta magát a hangja, de egyértelműen a hangszerrel való élő találkozás volt a legnagyobb élmény számomra. Ekkor született meg a fejemben az ötlet, hogy jó lenne a Rhodes hangjából készíteni egy VST plugint. Ennek segítségével nem kell a fizikai méreteit tekintve nem kicsi hangszert feltétlenül használni, hanem pusztán egy számítógép és egy Musical Instrument Digital Interface (MIDI) billentyűzet segítségével reprodukálható a hangja bárhol, bármikor és bármilyen hangerőn. Így szélesebb körben elérhető ennek a csodálatos hangszernek a hangja, akár stúdiófelvételeken vagy élő fellépéseken használható minőségben.

1.2. Története

A Fender Rhodes, eredetileg csak egyszerűen Rhodes, egy elektromoszongoratípus, amit az 1940-es, 50-es évek Amerikájában kezdett el kifejleszteni Harold Rhodes. Eredeti a célja a hangszerrel az volt, hogy hozzáférhetővé tegye a zongorázást, és ezáltal a zenélést, a háborúból visszatérő veteránok széles tömegének. Az eredeti változatok még lényegesen kisebb hangterjedelemmel rendelkeztek, mint a később sikeressé váló nagyobb modellek. A hangszer a közkedveltségét az 1960-as, 70-es években nyerte el. Ekkor lett elterjedt hangszer a rock és fusion stílusokban, olyan billentyűsöknek köszönhetően, mint Ray Manzarek a Doors-ból, vagy Herbie Hancock, aki többek között Miles Davis-szel is játszott, mielőtt sikeres szólóelőadóvá vált volna.

Az évek során a hangszernek számos változata jelent meg: például a Suitcase változat két beépített hangszóróval és egy hozzájuk tartozó erősítővel, valamint tremoló effekttel rendelkezik. A legelterjedtebb változat a 73 billentyűs Stage Piano. Ez egy egyszerűbb hangszer, az elektromos gitárokhoz hasonlóan külső erősítő szükséges hozzá.

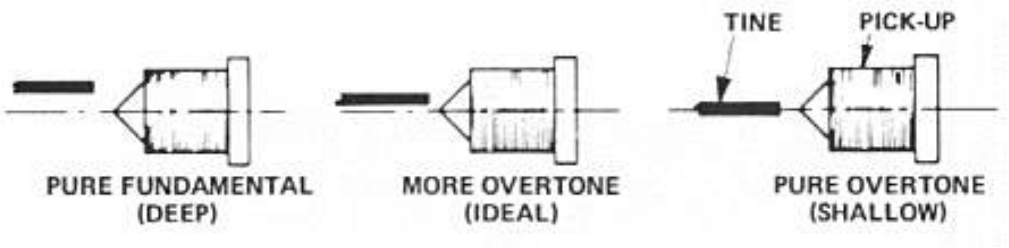
1. ábra. Rhodes zongora eltávolított fedőlappal

1.3. Hangképzése

A hangszer felépítése nagyon hasonlít a hagyományos zongorához abban a tekintetben, hogy itt is kalapácsok ütik meg a hangot biztosító rezgő közeget, viszont itt a zongora húrjaival szemben, *tine* elnevezésű fémlapok biztosítják a hangot (az 1. ábra). A tine-ok úgynevezett tone generator-okhoz csatlakoznak, amik hangvillaként funkcionálnak, a tine rezgését hosszabítják meg. A tine-ok hangolhatóak is a hozzájuk tartozó hangolórugóval. A fémlapok hangját pickupok veszik fel, egy elektromos gitárhoz hasonlóan. Ez az elektromos jel vezethető ki az erősítőre, és így kapható a Rhodes jellegzetes hangja.

Széles körben elterjedt különböző effektpedálok használata is a hangszerrel, szokás a Rhodes és az erősítő közé torzítópedált (distortion-t, vagy overdrive-ot) tenni, de a különböző modulációs effektek, mint a chorus, vagy a Suitcase változatba be is épített tremoló is gyakran alkalmazott a hangszeren.

Mivel nem teljesen szimmetrikusan helyezkednek el a tine-ok a hangszedőkhöz képest, ezért nem teljesen lineáris rendszerként írható le az átvitel, ebből kifolyólag jelennek meg felhangok a az eredményként kapott hangban. Ezek aránya a 2. ábrán látható módon változik az elhelyezkedés függvényében.



2. ábra. A hangszedő pozícójának kapcsolata a felhangok megjelenésével (kép forrása:[1])

A hangszedők és a tine-ok egymáshoz viszonyított pozíciója erősen befolyásolja a hang karakterét. Amennyiben egy vonalban helyezkednek el, akkor csak a felhangokat kapjuk meg, ha jóval lejjebb van a hangszedő, akkor pedig csak az alaphangot kapjuk. Az ideális elhelyezés ebből adódóan a kettő között van.

A hangszer egy fontos tartozéka még egy, a zongoráról ismerős zengető pedál *(sustain pedal)*, ami felemeli a filctompítókat a tine-okról, és így a kapott hang hosszú lecsengéssel rendelkezik, szabadon rezeg a tine.

1.4. A dolgozat célja

A dolgozat célja egy Virtual Studio Technology (VST) szoftverhangszer elkészítése, úgy, hogy az egy valódi Rhodes hangszer hangmintái alapján, az azokból kinyert, felhangokat jellemző frekvencia, amplitúdó és lecsengési idő adatok felhasználásával készíti el valós időben a hangmintákat, az adott hangmagassággal és leüteserősséggel rendelkező hanghoz.

2. A szoftverhangszer elkészítésének módja

A szoftverhangszer megvalósításában a megszólaltatáshoz szükséges hangminták előállításának két alapvetően különböző módja van: a valós idejű jelszintézis és a hangminta alapú szintézis. Az általam megvalósított VST hangszer az előbbit használja.

2.1. Valós idejű jelszintézis

A valós idejű hangszintézisnek az a nagy előnye a hangmintaalapú megvalósítással *(sampler)* szemben, hogy nem kell ebben az esetben nagyon nagy adatbázist felépíteni a hangmintákból. A sampler-nél szükséges, hogy mind a rendelkezére álló 73 hangból legyen több különböző leütéserősségű hangminta. Az ehhez tartozó hangminták elkészítése egyfelől nagyon hosszú időt venne igénybe, másrészt pedig az ilyen eljárást használó szoftverhangszer nagy RAM-igénnyel rendelkezik, hiszen a használt hangmintákat be kell tölteni a memóriába. Ezzel szemben a valós idejű hangszintézis során a szükséges hangmintát valós időben állítjuk elő, egy előre megalkotott modell segítségével, így egyszerűbb modell esetén könnyebben megvalósítható a szoftver. A modellezés során két alapvetően különböző módon lehet elindulni:

- a rendszer fizikájának modellezése – ekkor a hangszer hangképzésének fizikai oldala felől közelítjük meg a modellalkotást, fizikai modell segítségével szimuláljuk a keletkező hangot. Ezt a módszert fizikai alapú hangszintézisnek szokás nevezni.

- meglévő minták felhasználásával – ekkor rendelkezünk a hangszer hangjáról részletes hangfelvételekkel és ezeket elemezzük. Ezek segítségével határozunk meg egy modellt, amellyel leírható a hangszer hangképzése. Ezt az eljárást nevezzük jelminta alapú szintézisnek. A jelen esetben ezt az utat választottuk, mivel az 1.3. fejezetben tárgyalt nemlinearitások miatt nehezebb a fizikai modellezés. Továbbá rendelkezésre állt egy valódi hangszer, így a szükséges felvételek elkészíthetőek voltak.

Ennek a megközelítésnek az a hátránya, hogy míg ez a modell egy bizonyos közelítéssel él, addig a sampler-nél a hangszer valódi hangját halljuk, így az valósághűbb megszólalást kínál.

2.2. Digital Audio Workstation

A Digital Audio Workstation (DAW) egy hangfelvételek készítésére, szerkesztésére és manipulálására alkalmas számítógépes szoftver, amely központi szerepet játszik különféle különálló szoftverek, például VST-k, összekötésében. A program általában tartalmaz egy olyan felületet, amin belül végezhető a felvételek szerkesztése, valamint soksávos keverése. Ezen kívül általában rendelkezik beépített effektekkel és virtuális hangszerekkel, ezek használatára szintén lehetőséget biztosít. A modern hangfelvételek készítésének elengedhetetlen eleme egy ilyen szoftver, széles körben használt DAW például a *Pro Tools*, a *Cubase*, az *Ableton Live*, vagy az általam is

használt *Reaper*. Azért esett erre a választásom, mert ingyenesen elérhető, és viszonylag sok tapasztalatom van a használatával.

2.3. VST keretrendszer

A VST-Virtual Studio Technology, egy a Steinberg cég által 1996-ban kifejlesztett technológia, amely segítségével különféle szoftveres audióeffektek és virtuális hangszerek használhatóak egy számítógépen futó DAW-on belül. Ezen belül megkülönböztetünk VST hangszereket(VST Instrument-VSTi) és VST effekteket (VST Effect-VSTfx), a VSTi-k MIDI jelekből állítanak elő digitális hangmintákat, a VSTfx-ek pedig audiójelet manipulálnak. Az előbbi lehet virtuális hangszer alapú, vagy akár sampler is.[2]

2.4. MIDI szabvány

A MIDI szabvány azért készült el az 1980-as évek elején, hogy a különféle gyártók szintetizátorait, samplereit és stúdióeszközeit szabványos módon össze lehessen kötni egymással. A MIDI megjelenése előtt a gyártók különböző csatlakozókkal és protokollokkal tették lehetővé a saját gyártású eszközeik összekötését, viszont ez nagyon nagy akadály volt a zenészek számára. Ezt a problémát küszöböli ki a MIDI [3]. Ez fizikai szinten egy aszinkron soros vonali protokoll, egy 5 tüskéjű csatlakozó segítségével csatlakozik az összekapcsolandó eszközökhöz a kábel, ezen keresztül közvetíti a jeleket. Egy kábel segítségével egyszerre 16 különböző eszközt lehet vezérelni, ezek külön csatornaként jelennek meg a MIDI üzenetekben. A jelek a következő információkat közvetítik:

- a hang magassága,
- a hang leütéserőssége,

- a MIDI-csatorna, amelyre vonatkozik,
- az üzenet típusa.

Az üzenet 3 byte-ból áll, az elsőben, a státuszjelző byte-ban van az első 4 biten az üzenet típusa kódolva, a második négyen pedig a MIDI-csatorna száma. A másik két byte-ban az adat van eltárolva, a második byte-on van a hangmagasság kódolva, a harmadikon pedig a leütéserősség *(velocity)*. A MIDI-hangok magassága 0 és 127 között változhat egész számként, a legmélyebb 0-ás hang a zenei C_{-1} (8.18 Hz) hangnak felel meg, a legmagasabb 127-es pedig a zenei G_9-nak (12543.85 Hz), amennyiben 440 Hz-re van hangolva az A_4 hang. A leütéserősség szintén 0 és 127 között változhat annak megfelelően, hogy milyen erősen ütötték le a billentyűt, 0 amikor nincsen leütve a billentyű és 127 pedig a legerősebb leütés. A csatornaszám azt jelzi, hogy melyik MIDI-csatornára vonatkozik az információ, 0–15 között számozva. Az üzenet típusa pedig arra vonatkozik, hogy milyen típusú eseményt kódol az üzenet, például egy hang leütésére vonatkozik a *note-on* üzenet, a felengedésére pedig a *note-off*. Ezen kívül a *Program Change* üzenet jelezi a programváltást, de a különböző potméterek, például a hangmagasság változtatására szolgáló *mod wheel* változásáról is készülnek üzenetek, ezek a *Control Change* üzenetek [4].

2.5. Hanganalízis

Az analízishez használt kódot MATLAB-ban készítettem el. Azért erre a környezetre esett a választásom, mert már korábban is használtam, így viszonylag otthonosan mozgok benne. Emellett a MATLAB sok előre implementált függvénnyel rendelkezik, amelyek elengedhetetlenek az analízis elvégzéséhez. Így például nem kell szűrők karakterisztikáját implementálni, mert ez már el van előre készítve. Az elemzést végző program több nagyobb

egységből áll össze:

- *analyzer.m* – az analízis központi eleme, ennek segítségével lehet meghatározni az adott hangminta felharmonikusainak az $A_{\max}$ maximális amplitúdóját, τ időállandóját és az f_{n} frekvenciáját.
- *fund_freq.m* – ennek a függvénynek a segítségével lehet meghatározni a pontos alapfrekvenciát az *analyzer.m* függvényben, a benne implementált módszereket a 4.2. fejezet fejti ki részletesen.
- *hps.m* – a 4.2.2. fejezetben részletezett Harmonic Product Spectrum (HPS) módszert implementáló függvény.
- *rms.m* – a futó Root Mean Square (RMS)-t számító függvény, a *fund_freq.m*-ben szükséges a megfelelő jelszintű rész kikereséséhez.

2.6. Hangszintézis

A hangminták analízise után, az ebből kinyert információ alapján modellt illesztünk a rendszerre. A modellillesztésre is MATLAB környezetben került sor, a *polynomgen.m* fájlban található kód segítségével. Az eljárás a következő:

- A program először a hangfelvételeket beolvassa, ezekre meghívja az *analyzer* függvényt, így kinyerve belőlük az amplitúdó és időállandó információkat.
- Az ebből kinyert adatokra polinomokat illeszt, minden hang minden leütéserősségéhez, az összes felharmonikus amplitúdóira és időállandóira is.

- Végül az amplitúdókra és indőállandókra illesztett polinom segítségével extrapolációval megbecsüljük a Rhodes két legszélső hangjára, az E_1-re és az E_7-re jellemző amplitúdó- és időállandóértékeket. Erre azért van szükség, hogy a későbbiekben az amplitúdó számítása során 2 dimenzióban csak interpolálni kelljen, és ne legyen szükség további extrapolációra. Valamint az időállandóknál is csak az interpolálás műveletét kelljen implementálni.

- A kapott mátrixokat a program külön-külön szövegfájlokba írja, hogy a későbbiekben onnan tudja kiolvasni a plugin.

Ha a felhasználó leüt egy billentyűt, az meghatároz egy adott MIDI-hangot 0 és 127 között, ami a hangmagasságot fogja mutatni. A leütéserősség *(velocity)* pedig abból fog származni, hogy milyen erősséggel ütötte le a felhasználó az adott billentyűt. Ezen adatok alapján el lehet végezni az előbbiekben megkapott polinomok segítségével az interpolációt, vagy bizonyos esetekben az időállandókra az extrapolációt.

2.7. VST plugin implementálása

A VST plugin implemantálása a Steinberg honlapjáról ingyenesen letölthető VST 3 Software Development Kit (SDK) segítségével történt. Ebben egy C Application Programming Interface (API)-n alapuló de C++ osztályokat tartalmazó csomag található. Így a további kódolás Microsoft Visual Studio-ban történt meg C++ nyelven. Az általam fejlesztett szoftverhangszer alapját a csomagban található *Note Expression Synth* VSTi jelentette. Ennek tanulmányozásával és módosításával alakult ki a végső szoftver. A megvalósításhoz továbbá implementálni kellett a MATLAB *interp1*, egydimenziós és a *interp2* kétdimenziós interpoláló függvényét C++ nyelven,

tekintve, hogy központi szerepet játszanak a szintézis megvalósításában.

3. Hangfelvételek készítése

3.1. A rögzített hangszer

A felvételeket az 3. ábrán látható, 1973–75-ből származó Mark I-es Rhodes Stage Piano-val készítettem, ez a változat egy 73 billentyűs hangszer. A zongora nagyon jó állapotban maradt mind a mai napig, a mindennapos használat ellenére, viszont pont a rendszeres használat miatt a hangszer megfelelően be van állítva.

3. ábra. A rögzített hangszer

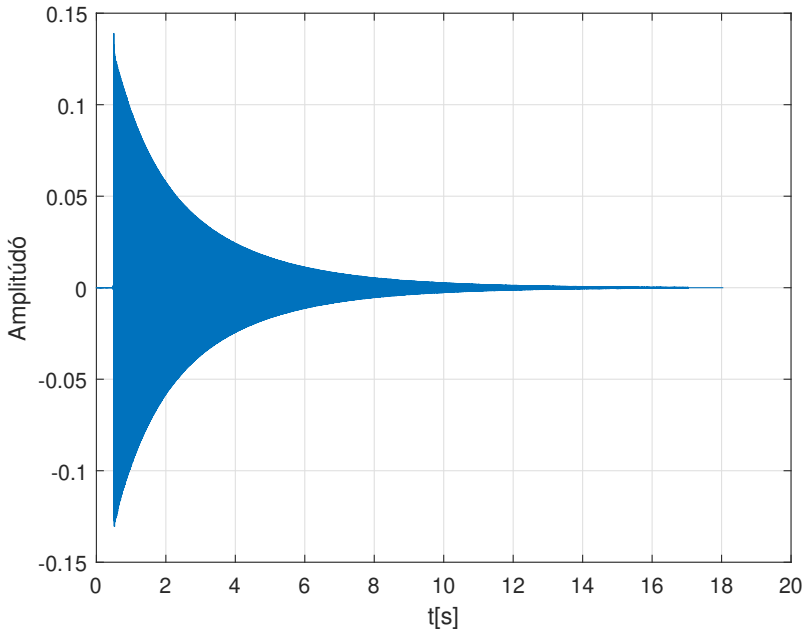
3.2. A felvételi eljárás

A felvételi lánc a következőképpen állt össze: a hangszerből egy 6.3-as jack kábellel jutott a jel a PreSonus Audiobox USB 96 külső hangkártya vonali bemenetére. A hangszeren a hangerő és a bass boost is 7-esre volt állítva a 10-ből, ez egy eléggé általános beállítás. A hangkártyán a jelszint úgy volt beállítva, hogy még ne vezérlődjön túl a bemenet a legerősebb leütésnél sem, viszont a leggyengébb is megfelelő nagyságú jelet generáljon. A felvételeket a Reaper Digital Audio Workstationnel készítettem el, ezzel is editáltam őket megfelelő hosszúságúra később. A zongorán az összes zenei A hangról készítettem felvételeket, A_1-től (55 Hz), A_6-ig (1220 Hz). A felvételi eljárás során egy hangról készítettem nagyon sok felvételt, különböző leütéserősségekkel: az alig hallhatótól, egészen a nagyon erős leütésig, és ebből válogattam ki végül 8 egymástól jól elkülöníthető dinamikájú hangot. Az eljárásom az volt, hogy egyre erősebben ütöttem le az adott billentyűt, figyelve arra, hogy az előzőleg leütött hang már teljesen lecsengjen, amire a bemenő jelszint -60 dB alá csökkenéséből következtettem. A DAW-ban nyomon követhető a felvett hangok időfüggvénye, így láthatóvá válik a leütéserősség. A hallható különbségek mellett ez nyújt segítséget a különböző leütéserősségű hangok kiválasztásánál. Ezt a műveletet megismételtem minden A hangra, és így kaptam az analízisre szolgáló hangmintákat. A felvételek 48 kHz-es mintavételi frekvenciával készültek, így nagyobb frekvenciatartományúak a felvételek, mintha a standard 44.1 kHz-en lennének felvéve, viszont nem nagy a fájlok mérete.

4. A hangminták elemzése MATLAB környezetben

4.1. Áttekintés

A hangszer hangképzéséből kiindulva, miszerint egy kalapács megüti az adott hanghoz tartozó tine-ot, leírható ez úgy, mint egy impulzussal gerjesztett rúd, amely a gerjesztés után magára lett hagyva. Ez a mechanikai rendszer a rezonanciafrekvenciáján fog rezegni, és a belső veszteségek miatt exponenciálisan lecseng a hang, amint az a 4. ábrán látható. A frekvanciatar-



4. ábra. Egy jellemző időfüggvény (az A_3 hang 4-es leütéserősséggel)

tományban megnézve a jelet láthatjuk, hogy a jel egy alaphangból és a hozzá tartozó felhangok rendszeréből áll. Általában az alaphangnak a legnagyobb az amplitúdója. A következő képlettel modellezhetünk egy ilyen típusú han-

got:

$$X(t) = \sum_{n=1}^{N} \sin(n \cdot 2\pi \cdot f_0 \cdot t) \cdot A_{\max_n} \cdot e^{-\frac{t}{\tau_n}} + \phi_n \tag{4.1}$$

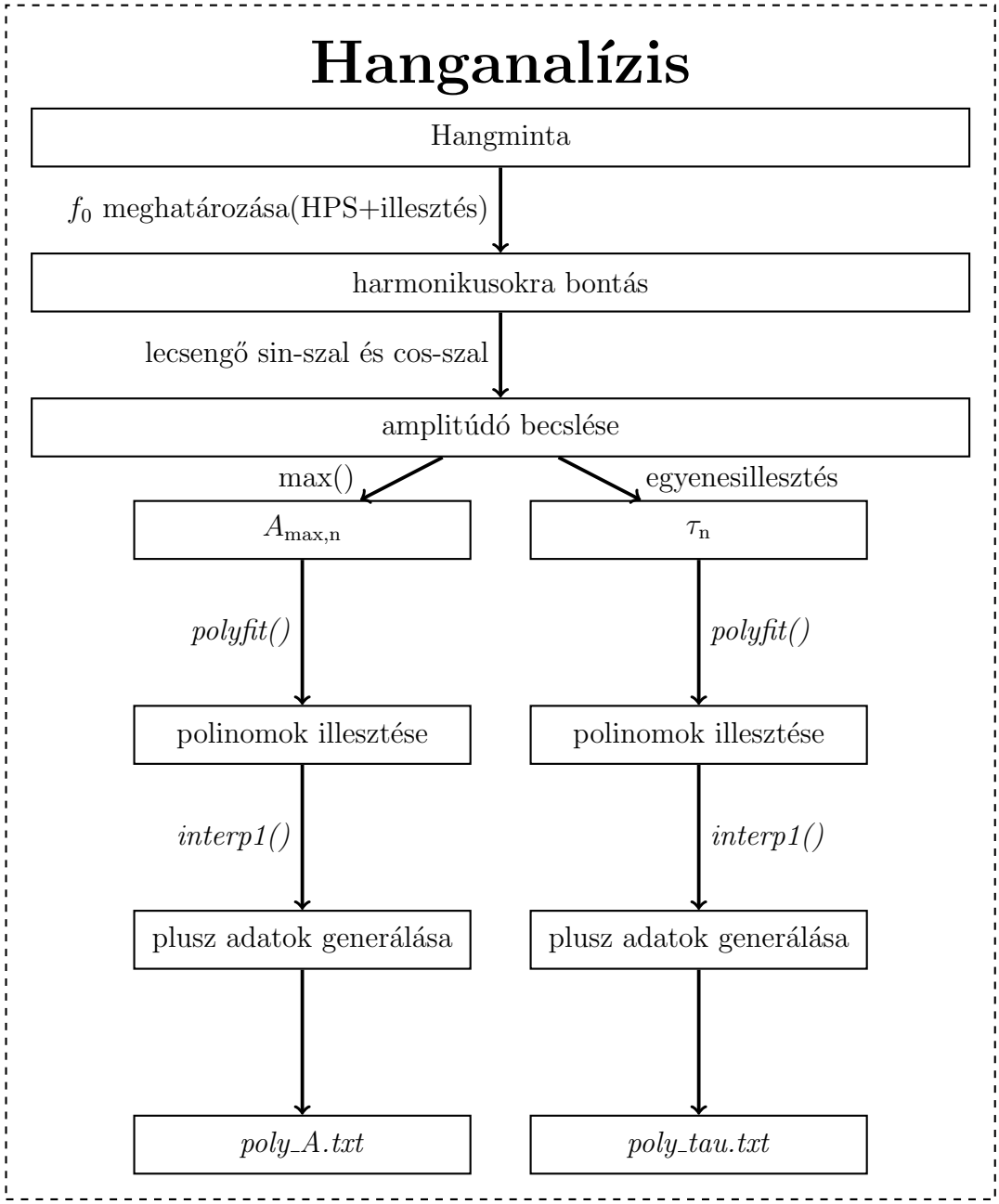
ahol:

- f_0 – alapfrekvencia;
- $A_{\max_n}$ – az n-edik felharmonikus maximális amplitúdója;
- τ_n – az n-edik felharmonikus exponenciális lecsengésének időállandója;
- N – a felhangok száma.
- Φ_n – az n-edik felharmonikushoz tartozó kezdőfázis

Ezen változók a meghatározása MATLAB környezetben történt, itt készítettem el a számításhoz szükséges kódot. Az 5. ábrán látható a teljes analízis folyamata.

4.2. Az alapfrekvencia meghatározása

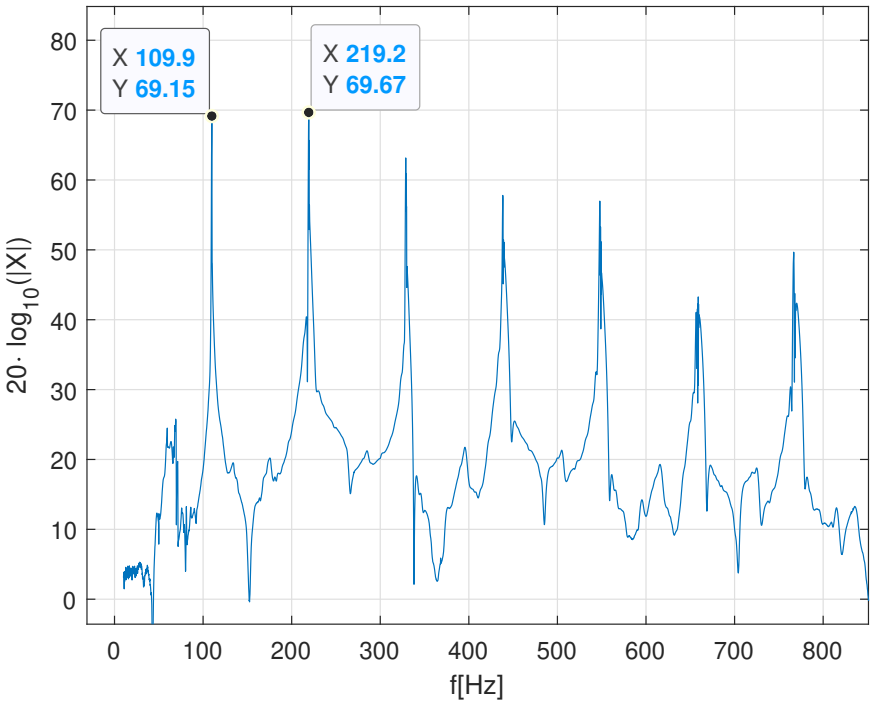
Az alapfrekvencia meghatározása elsőre egészen kézenfekvően megoldható problémának tűnik, a frekvenciatartományban vizsgálva a legnagyobb amplitúdóval rendelkező frekvenciát kiválasztjuk egy egyszerű maximumkereséssel, és így megkapható az alapfrekvencia. Ezzel az eljárással kapcsolatosan több probléma is felmerül. Egyrészt, semmi sem garantálja, hogy az alapharmonikusnak a legnagyobb az amplitúdója, és bár általában ez igaz, amint az a 6. ábrán is látható, néha a másodiknak nagyobb, és ilyen esetben ezt találnánk meg. Másrészt pedig a frekvenciaspektrumot a Fourier-transzformáció segítségével kaphatjuk meg. Viszont mivel az elemezni kívánt jel nem folytonos, hanem egy adott mintavételi frekvenciával diszkrét időpontokban mintavételezve lett, és nem is periodikus, ezért a



5. ábra. A hanganalízis folyamata, és a szintézis előkészítése MATLAB-ban

diszkrét Fourier-transzformációt, röviden DFT-t, kell használni. Ezzel csupán közelíteni lehet a jel spektrumát, a továbbiakban részletezett problémák mi-

att.



6. ábra. Frekvenciaspektrum a A_2 hang 7-es leütéserősségű hangmintájához

4.2.1. Fast Fourier Transform

A Discrete Fourier Transform (DFT) során a jelet egy T hosszúságú ablakfüggvénnyel ablakoljuk, és az ebbe eső jelszakasz periodikus kiterjesztésének a Fourier-sorát kapjuk. Amennyiben a jel nem periodikus T-ben, ablakfüggvényt kell alkalmazni annak érdekében, hogy a jel periodicizálása során ne keletkezzenek ugrások a jelben. Ennek következtében ugyanis a spektrumban olyan plusz komponensek jelennek meg, amelyek az eredeti jel spektrumában valójában nincsenek benne. Ezt a jelenséget nevezzük spektrális szivárgásnak. Az egyik elterjedt ablakfüggvény a Hann-ablak, ez

egy emelt koszinusz-görbe, amelynek a minimumai a két szélénél vannak. Így a jel értéke az ablak szélénél nullával lesz egyenlő, nem jelentkezik benne ugrás, így jól periodicizálható. Egy T hosszúságú periódusban N minta található, ezek a minták Δt távolságra vannak egymástól, ahol $\Delta t = \frac{1}{f_s}$, f_s a mintavételi frekvencia. Ebből a jel DFT-je a következő képlet szerint számítható:

$$X[k] = \sum_{n=0}^{N-1} u[n] e^{-\mathrm{j}kn\frac{2\pi}{N}}, \text{ ahol } k = 1, 2, \ldots N-1 \tag{4.2}$$

Ennek a műveletnek a hatékony számítására szolgál a Fast Fourier Transform (FFT) művelet, az eredetileg $O(N^2)$ komplexitású műveletet, így $O(N \log N)$ komplexitásúként lehet számítani.

Az így kapott spektrum felbontása véges, $\Delta f = \frac{f_s}{N}$, tehát függ az f_s mintavételi frekvenciától, valamint attól, hogy hány darab minta van egy adott ablakban (N), vagyis az ablakok hosszától, mivel a minták távolsága egymástól adott. Amennyiben az adott frekvenciaértékhez tartozó csúcs nem pont egy ilyen vonalra esik, akkor a transzformáció eredményeként nem megfelelő amplitúdóértéket kapunk itt, mivel az FFT művelet a jel teljesítményét teljes egészében megtartja. Ezt a jelenséget léckerítés hatásnak nevezzük, és periodikus jeleknél koherens mintavételezéssel küszöbölhető ki. Tehát egyszerűen az FFT és az ezen való maximumkeresés alkalmazásával nem tudjuk pontosan meghatározni a jel alapfrekvenciáját, a véges frevenciafelbontás miatt.

4.2.2. Harmonic Product Spectrum

A nem legnagyobb amplitúdójú alapfrekvencia problémáját küszöböli ki a HPS módszer, amely a felhangrendszer azon tulajdonságát használja ki, hogy a felhangok frekvenciái az alapfrekvencia egész számú többszörösei.

A módszer matematikailag a következő módon írható fel [5]:

$$X_{\mathrm{HPS}}(f) = \prod_{n=1}^{N} |X(n \cdot f)| \tag{4.3}$$

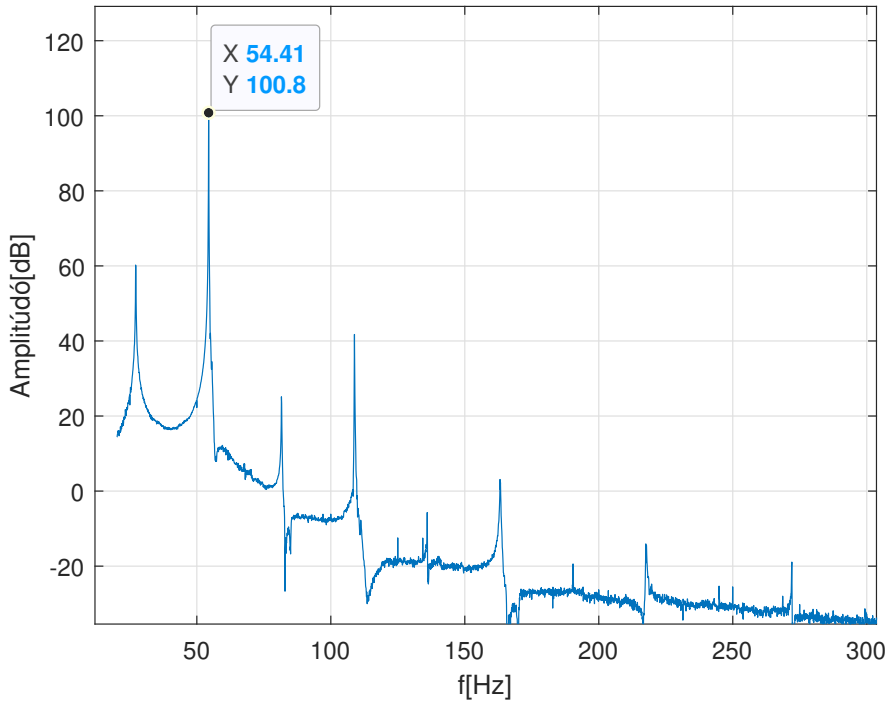
Tehát úgy kaphatjuk meg az $X_{\mathrm{HPS}}(f)$ spektrumot, hogy az eredeti jel $X(f)$ spektrumától kezdve, összeszorozzuk a jel minden n-edik komponensét tartalmazó spektrumot az előző lépésben kapott spektrummal. Így az n-edik iterációban, az $n-1$. felhang amplitúdója kerül az alapfrekvencia helyére, kvázi n-szeresen összenyomva a spektrumot az f tengely mentén. Ez a produktumképzés következtében az alapfrekvenciához tartozó amplitúdót növeli, ugyanennek köszönhetően a többi komponens vagy 0-val szorzódik, vagy pedig zajkomponensekkel, így azok amplitúdója drasztikusan lecsökken. Ha elégszer ismételjük a fenti eljárást, akkor biztosan kijelenthető lesz, hogy a legnagyobb amplitúdója az alaphanghoz tartozó komponensnek lesz (7. ábra), és így egy egyszerű maximumkereséssel már meg lehet azt találni:

$$\hat{X}(f_0) = \max_{f_n}(X_{HPS}(f_{\mathrm{n}})) \tag{4.4}$$

4.2.3. A valódi alapfrekvencia meghatározása

Így közeli becslést kaphatunk az alaphang frekvenciájára, viszont a HPS módszer sem ad teljesen pontos eredményt, az előzőekben részletezett felbontásbeli probléma továbbra is fennáll. A valódi alapfrekvencia megtalálásához meg kell határozni a valós és a becsült frekvenciák különbségét, majd ezzel kompenzálva a HPS segítségével kapott becslést, megkapható a valódi alapfrekvencia.

Az eredeti $x(t)$ jelünk valós, vagyis a spektruma komplex-konjugált szimmetrikus. A frekvenciaeltérés meghatározása a következőképpen zajlik [6].



7. ábra. Az A_1, 1-es leütéserősséggel vett hang spektruma *HPS* végzése után

Először egyoldali frekvenciamoduláljuk az eredeti jelet $\hat{f}_0$ becsült alapfrekvenciával:

$$x_{\text{mod}}(t) = x(t) \cdot e^{-2\pi \cdot \text{j} \cdot \hat{f}_0 \cdot t} \tag{4.5}$$

Ahol az x_{mod} jel egy komplex jel, melynek a spektruma az eredeti $x(t)$ jelé eltolva $\hat{f}_0$-el balra. Így a kapott spektrumban az f_0 alapharmonikus frekvenciájához tartozó érték a $\Delta f = f_0 - \hat{f}_0$ pozícióba kerül, ami a HPS pontossága miatt egy kis érték.

Ezt a jelet egy aluláteresztő szűrővel leszűrve csak a Δf frekvenciájú, A amplitúdójú, ϕ kezdőfázisú komponens marad, amit a következő módon írhatunk fel:

$$x_{\text{filt}}(t) = A \cdot e^{\text{j} \cdot (2\pi \cdot \Delta f \cdot t + \phi)} \tag{4.6}$$

Mivel az eredeti jel is csak f_s sűrűséggel mintavételezve áll rendelkezésre, ezért diszkrét időben a jel:

$$x_{\text{filt}}[k] = x_{\text{filt}}(k \cdot \Delta t), \text{ ahol } \Delta t = \frac{1}{f_{\text{s}}}$$

Így diszkrét időtartományban az x_{filt} jel:

$$x_{\text{filt}}[k] = A \cdot e^{\text{j}\cdot(2\pi\cdot\Delta f\cdot k\cdot\Delta t+\phi)} \tag{4.7}$$

Szükség van az x_{filt} jelre a $k+1$-edik időpillantban is, ennek segítségével tudjuk elvégezni az időtartománybeli illesztést, amivel meg tudjuk határozni a Δf értékét.

$$x_{\text{filt}}[k+1] = A \cdot e^{\text{j}\cdot(2\pi\cdot\Delta f\cdot(k+1)\cdot\Delta t+\phi)} \tag{4.8}$$

A két egymást követő érték hányadosa:

$$\frac{x_{\text{filt}}[k+1]}{x_{\text{filt}}[k]} = e^{\text{j}\cdot 2\pi\cdot\Delta f\cdot\Delta t} \tag{4.9}$$

A hányados értéke nem függ k-tól, viszont csak abban az esetben konstans az értéke, ha tökéletesen harmonikus a jelünk, ez a jelen esetben nem áll fent, mivel zajjal terhelt az aluláteresztő szűrővel szűrt jel. Ennek kiküszöbölésére vezessük be a γ konstanst, és ennek segítségével írjuk fel a túlhatározott egyenletrendszert minden mintára:

$$\begin{bmatrix} x_{\text{filt}}[k+1] \\ x_{\text{filt}}[k+2] \\ \vdots \\ x_{\text{filt}}[N] \end{bmatrix} = \gamma \begin{bmatrix} x_{\text{filt}}[k] \\ x_{\text{filt}}[k+1] \\ \vdots \\ x_{\text{filt}}[N-1] \end{bmatrix}, \text{ ahol } N = \text{a minták száma} \tag{4.10}$$

Az egyenletrendszert például a legkisebb négyzetek módszerével megoldva megkapjuk a γ értékét. Ebből pedig a

$$\Delta f = \frac{\arg(\gamma)\cdot f_{\text{s}}}{2\pi} \tag{4.11}$$

képlet segítségével, kapjuk meg a Δf értékét. Ezzel pedig a valódi alapfrekvencia egyszerűen megkapható:

$$f_0 = \hat{f}_0 + \Delta f. \tag{4.12}$$

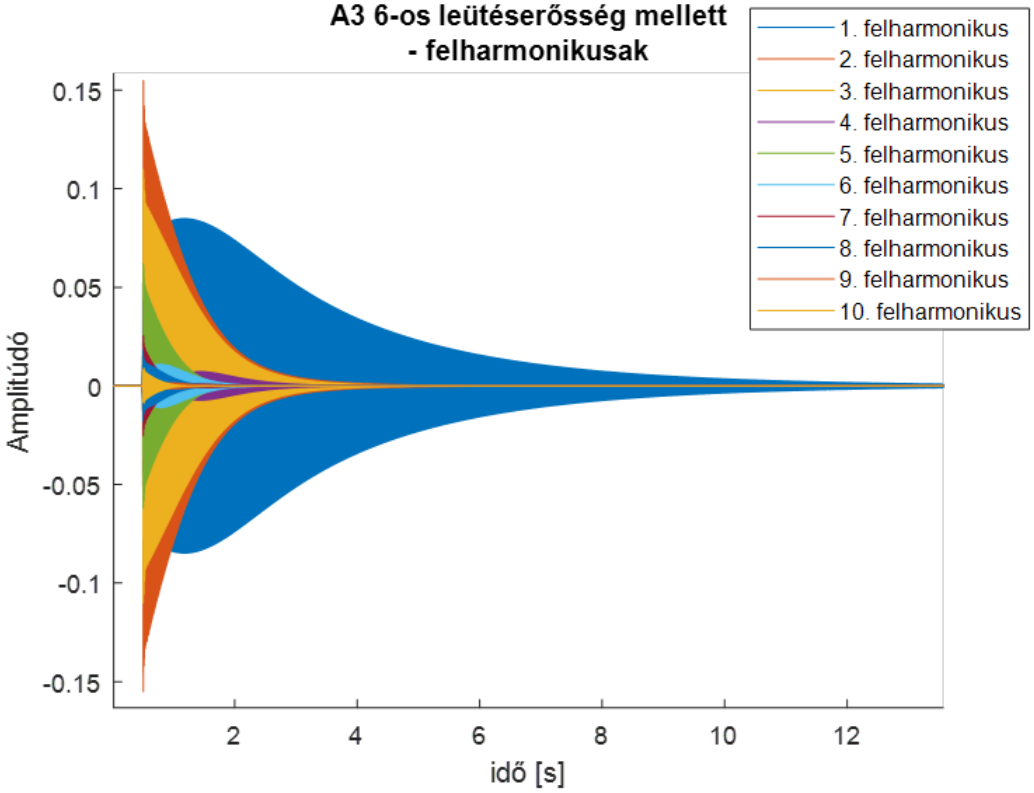
Ez alapján az n-edik felharmonikus frekvenciája:

$$f_{\mathrm{harm_n}} = n \cdot f_0. \tag{4.13}$$

4.3. Harmonikusokra bontás

Az f_0 ismeretében, harmonikusaira tudjuk már bontani az eredeti hangmintánkat, hogy ezekre külön-külön meg tudjuk határozni a rájuk jellemző $A_{\max}$ maximális amplitúdókat, és a τ időállandókat. Kísérletezés útján arra jutottunk, hogy 10 darab harmonikussal már valósághűen visszaadhatóak a hangok. Ennél nagyobb értéknél már túl alacsony lesz az amplitúdója a felharmonikusoknak, így nem lehet őket megbízhatóan elemezni. Az alacsony amplitúdó miatt az eredeti jelben sem hallhatóak.

A harmonikusokra bontást az adott harmonikusra szűrő sávszűrővel lehet megoldani, a felbontás után a különböző harmonikusok időfüggvénye külön vizsgálható. A szűrő karakterisztikájának meghatározásánál figyelembe kell venni, hogy a szűrő sávszélessége elég nagy legyen. Ennek érdekében meghatároztunk egy minimális 100 Hz-es sávszélességet, amennyiben ez nagyobb mint az alapfrekvencia harmada. Valamint az alsó határfrekvenciának 20 Hz-et, ha ennél alacsonyabb jönnek ki a sávszélesség alapján. Ezeknek a korlátozásoknak köszönhetően az így meghatározott karakterisztikájú szűrők stabilak maradnak az összes megjelenő harmonikusra. Ezen határfrekvenciák segítségével készítünk egy harmadfokú Butterworth-szűrőt, a MATLAB beépített *butter()* függvényével. Ennek segítségével szűrjük ki az adott har-



8. ábra. Az A_3 hang felharonikusokra bontva, 6-os leütéserősség mellett

monikust a *filtfilt()* függvény segítségével, amely zérus fázistolással végzi el a szűrést.

4.4. A harmonikusok maximális amplitúdójának meghatározása

Innentől kezdve a jelet blokkonként dolgozzuk fel, melynek során a leszűrt harmonikusok időfüggvényeit adott számú mintából álló blokkokra bontottam. A blokkhossz meghatározásánál nagyon fontos figyelembe venni a jel alapfrekvenciáját. A blokkméret ne legyen túl kicsi, hogy ebből legalább egy periódus beleférjen a blokk időablakába, ellenkező esetben nem kapunk meg-

felelő eredményt. Viszont az sem jó, hogyha túlságosan nagyra választjuk a blokkméretet, mert akkor a felbontás nem lesz elég kicsi, és ez hamisítja meg az eredményeket. Ezekből következően az ideális blokkméret 1024 minta lett, 256 mintányi átfedéssel.

A harmonikusokat jellemző maximális amplitúdók meghatározáshoz először a 4.2 fejezetben tárgyalt, frekvenciabecslés hibájának számításához használt módszer segítségével ki kell számítani az adott blokkra vonatkozó frekvenciaeltérést az eredetileg számolt frekvenciához képest. Erre azért van szükség, mert az amplitúdó meghatározását úgy végezzük, hogy egy lecsengő szinusz és koszinusz összegével becsüljük a jel burkolóját, és ezek előállításához szükség van a pontos frekvenciára. A jelünk ugyanis a következő formában írható fel:

$$P \cdot \sin(\omega t + \Phi) \cdot e^{-\frac{t}{\tau}} = (A \cdot \cos(\omega t) + B \cdot \sin(\omega t)) \cdot e^{-\frac{t}{\tau}}, \tag{4.14}$$

ebből kifejezve:

$$\Phi = \arctan\left(\frac{A}{B}\right) \tag{4.15}$$

valamint:

$$P = \sqrt{A^2 + B^2} \tag{4.16}$$

Tehát az amplitúdó burkolója a Φ fázistolástól nem lineáris módon függ, ezért az előző bekezdésben említett módon lecsengő szinusz és koszinusz segítségével becsüljük azt. A 4.10. egyenletből ismert módon a túlhatározott egyenletrendszerből meghatározzuk a γ paramétert, ennek segítségével:

$$\Delta f = \frac{\arg(\gamma) \cdot f_s}{2\pi}, \text{ valamint} \tag{4.17}$$

$$\alpha = \log(|\gamma|) \cdot f_s = -\frac{1}{\tau}, \tag{4.18}$$

ahol az α az erősítés Ezekből a becslő lecsengő szinusz és koszinusz:

$$s = e^{\alpha \cdot t_{\text{blokk}}} \cdot \sin(2\pi \cdot (f_{\text{harm}} + \Delta f) \cdot t_{\text{blokk}}), \tag{4.19}$$

$$c = e^{\alpha \cdot t_{\text{blokk}}} \cdot \cos(2\pi \cdot (f_{\text{harm}} + \Delta f) \cdot t_{\text{blokk}}), \tag{4.20}$$

ahol $f_{\text{harm}} = n \cdot f_0$ az adott felhanghoz tartozó frekvencia és ezt kell még korrigálni az adott időablakban tapasztalható frekvenciaeltolódással (Δf). A t_{blokk} pedig a blokkhoz tartozó időtengely. Ezek segítségével felírható egy túlhatározott egyenletrendszer az $x_{\text{filt}}[k]$-ra:

$$\begin{bmatrix} x_{\text{filt}}[0] \\ x_{\text{filt}}[1] \\ \vdots \\ x_{\text{filt}}[N] \end{bmatrix} = \begin{bmatrix} s[0] & c[0] \\ s[1] & c[1] \\ \vdots & \vdots \\ s[N] & c[N] \end{bmatrix} \times \begin{bmatrix} w_s \\ w_c \end{bmatrix}, \tag{4.21}$$

ahol az x_{filt} a harmonikusra sávszűrt eredeti jel. Ebből meghatározható a jel amplitúdója, az A_{filt}:

$$|A_{\text{filt}}| = \sqrt{w_{\text{s}}^2 + w_{\text{c}}^2} \tag{4.22}$$

Ennek segítségével a blokk közepén becsülhető az amplitúdó a szinusz és a koszinusz együtthatóinak négyzetösszegének a gyökével:

$$X_{amp} = \left| x_{\text{filt}} \left[\frac{l_{\text{blokk}}}{2} \right] \right| = |A_{\text{filt}}| \cdot e^{-\frac{l_{\text{blokk}}}{2\tau}}, \tag{4.23}$$

ahol az l_{blokk} a blokk hossza minták számában, vagyis az ablak közepén vesszük az értéket. Ennek segítségével könnyen meghatározható a maximális amplitúdó az n-edik adott harmonikusra, egy egyszerű maximumkereséssel:

$$A_{\text{max,n}} = \max(X_{\text{amp,n}}) \tag{4.24}$$

Mivel a jel 24 bites bitmélységgel készült, ezért meghatároztuk egy abszolút minimumértéket, ami alatt 0-nak vesszük az amplitúdót:

$$A_{\text{max}_{\text{min}}} = \frac{1}{2^{23}} \tag{4.25}$$

Ez a minimális amplitúdó, amit a 24 bites bitmélység lehetővé tesz. Ez alatt már csak a kvantálási zajra illesztenénk, ezért 0 amplitúdót rendelünk hozzá, a felvétel zaja még ennél is nagyobb.

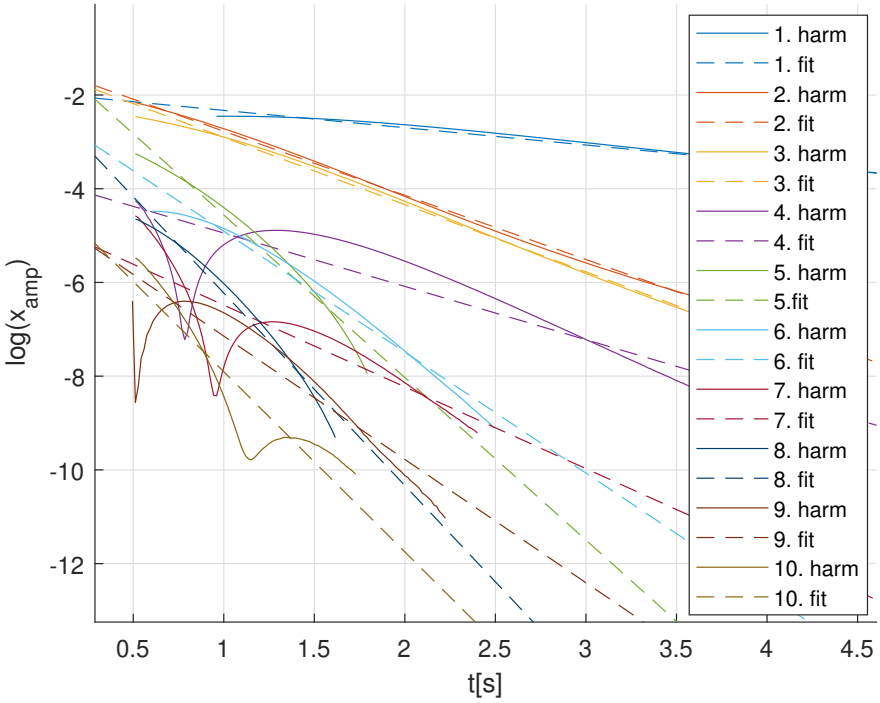
4.5. A τ_{n} időállandók meghatározása

Az időállandók meghatározása a blokkos feldolgozásból származó amplitúdóadatok segítségével történik. Amint azt az előzőekben láthattuk, a harmonikusok általánosan exponenciálisan csengenek le, és ezt használjuk ki az illesztés során. Először meg kell határozni azt az időintervallumot, ahol a jel amplitúdója elég nagy. Több próbálkozás után arra jutottunk, hogy a legjobb eredményeket akkor kapjuk, ha a maximális ampiltúdóértéktől kezdve vesszük a jelet egészen ennek az értéknek a századáig. Ha nem cseng le túl gyorsan ez a harmonikus, több mint három blokkban is az előzőekben meghatározott kritériumoknak megfelelő értékű az amplitúdó, és $A_{\mathrm{max}_{\mathrm{harm}}} \geq A_{\mathrm{max}_1} \cdot 0.005$. Mivel exponenciális a lecsengés, ha az amplitúdó burkolóját természetes alapú logaritmikus skálán ábrázoljuk, akkor egyszerű egyenesillesztéssel meg lehet határozni az időállandókat, az egyenesek meredekségeként. Az illesztéshez a MATLAB *polyfit()* függvényét használjuk, és ennek segítségével illesztünk elsőfokú polinomot a burkológörbe logaritmusára. Az n-edik harmonikushoz tartozó τ_{n} időállandó ebből a következőképpen határozható meg:

$$\tau_n = -\frac{1}{p_1}, \tag{4.26}$$

ahol a p az illesztett polinom polinomegyütthatóinak vektora, ebből az első a legmagasabb fokszám együtthatója, jelen esetben az x^1-n kitevője.

A 9. ábrán folytonos vonallal láthatóak a burkológörbék logaritmikus skálán, szaggatottal pedig a rájuk illesztett egyenesek. Az illesztés jól közelíti



9. ábra. Időállandók becslésének folyamata az A_3 hang 5-ös leütéserősségére

azokat a harmonikusokat, amelyeknek tisztán exponenciális a lecsengése, viszont vannak olyan harmonikusok is,amelyek lecsengésében látható egy lokális minimum. Ezekre kevéssé illeszkedik jól az egyenes, mivel nem tisztán exponenciális a lecsengés. Viszont az így kapott értékekkel is egy meghallgatásos összehasonlítással nagyon hasonló hangot kapunk, és ez a szoftver célja. Ennek az az oka, hogy ezeknek a harmonikusoknak általában jóval kisebb az amplitúdója, mint a legerősebb komponensé, és így az elnyomja őket.

Az illesztés során kijöhetnek negatív τ-k az illesztés hibájából kifolyólag, amennyiben nem megfelelően választjuk meg a blokkméretet. Az ilyen időállandó nem is fedi le a valóságot, amint a 9. ábrán is látható, bizonyos harmonikusoknál van egy felfutási szakasz, vagy elhalkulás, aztán felhangosodás,

majd megint lehalkulás, de az alapvető tendencia csökkenő. Ez a 4.1. fejezetben részletezett fizikai modell segítségével könnyen értelmezhető, hiszen egy magára hagyott rendszerről van szó, mely az impulzus után visszakerül nyugalmi állapotba. Így a negatív időállandó kiszűrésére ezek értékét Not a Number (NaN)-ra állítjuk, a hozzájuk tartozó maximális amplitúdók értékét pedig 0-ra . Erre azért van szükség, mert a negatív időállandók az 5.1.1. fejezetben részletezett polinomillesztés során nagyon meghamisítják az illesztés eredményét. Viszont a tévesen negatív τ-k illesztése általában elkerülhető a blokkméret helyes beállításával.

5. A hangszintézis előkészítése

A szintézis megvalósításához a következő lépéseket kell elvégezni:

- **Polinomok illesztése** – Az elemzésből kinyert amplitúdó és időállandó adatokra polinomot illesztünk, így egyszerűsítve az adatok feldolgozását és az interpolálást.

- **További adatok generálása** – Az elemzett hangminták nem tartalmazzák a Rhodes legmélyebb és legmagasabb hangját. Így ahhoz, hogy a VST hangszer megvalósításában az össze hanghoz tartozó adat meghatározható legyen interpolálással, a szélső hangokhoz tartozó adatokat meg kell határoznunk extrapolálással.

- **Az interpoláló algoritmusok implementálása C++ nyelven** – Egy tetszőleges leütéserősségű és hangmagasságú hang felharmonikusait leíró mérőszámokat az analízisből kinyert adatok alapján interpolálással kapjuk meg. A virtuális hangszert C++ környezetben implementáltam, ezért ezen a nyelven kell megvalósítanunk egy- és

kétdimenziós interpoláló algoritmusokat.

- **A kívánt hangminta összeállítása harmonikusonként** – Az interpolációval megkapott amplitúdó- illetve időállandóértékek segítségével az additív szintézis szerint felharmonikusonként állítjuk össze a végleges hangmintát.

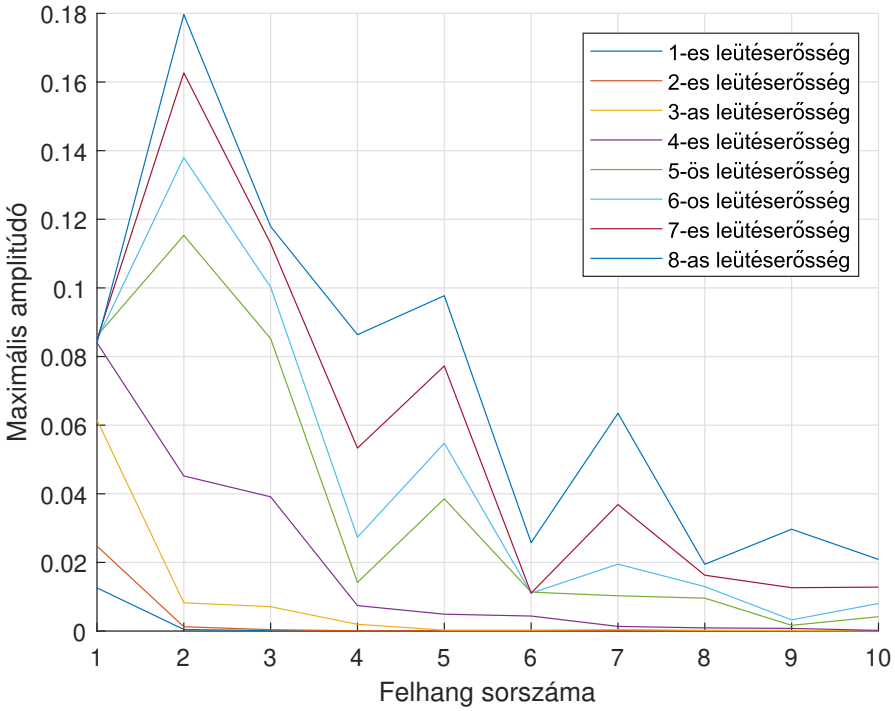
5.1. Polinomok illesztése

Az összes rendelkezésre álló hangfelvételre elvégzett analízisből megkapjuk az összes hangminta összes elemzett felharmonikusára a maximális amplitúdók és az időállandók értékét. Ezek alapján szeretnénk olyan függvényeket meghatározni rájuk, amelyek jól jellemzik a viselkedésüket, és ennek köszönhetően kevésbé lesznek kiugró értékek az adatok között, így az interpolálással jobb eredményeket kaphatunk. Az illesztést is MATLAB-ban végezzük a *polyfit()* függvény segítségével. Ennek meg lehet adni az illeszteni kívánt polinom fokszámát, így széles körben használható.

A 10. és a 11. ábrán megfigyelhető, hogy az egymást követő felhangokhoz tartozó értékek között viszonylag nagy változás tartozik, így érdemesebb a páratlan és páros felhangokra külön-külön elvégezni az illesztést. [7] Ha megnézzük, így már jóval kisebb változásokat kell lekövetnie az illesztett függvénynek. Ez látható például a 12. ábrán.

5.1.1. Polinomok illesztése az amplitúdókra

A maximális amplitúdókra való illesztésnél az előzőeknek megfelelően szétválasztjuk a páros és páratlan sorszámú harmonikusokat. Az illesztés minden felvett hang, minden leütéserősség mellett keletkező harmonikusaira történik, így ciklikussá tehető az illesztés folyamata. Az illesztést nem

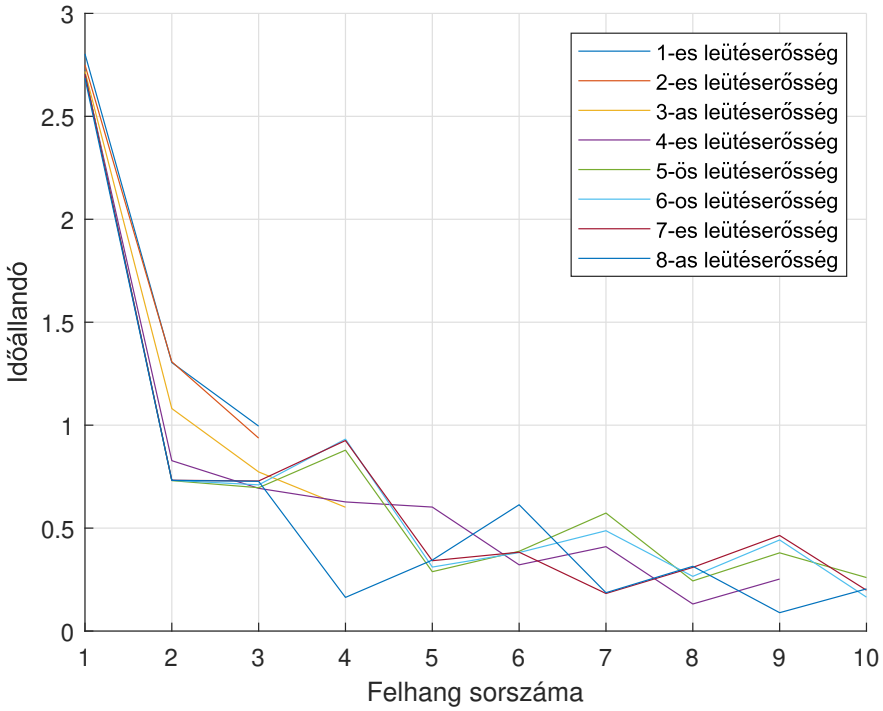


10. ábra. A_3 hang felhangjainak maximális amplitúdói

egyszerűen a maximális amplitúdóértékekre végeztük el, hanem először lenormáltuk az alaphang amplitúdójával, így az összes görbe 1-ből indul. Az így kapott értéknek a logaritmusára illesztettünk végül egy harmadfokú polinomot. A természetes alapú logaritmus miatt az összes görbe 0-ból indul így, valamint az amplitúdóértékek is kisebbek lesznek, így könnyebb a görbeillesztést kivitelezni. Ez meg is figyelhető a 13. ábrán, ahol a teli vonallal a mért maximális amplitúdók vannak ábrázolva a különböző leütéserősségekre, míg szaggatottal a rájuk illesztett polinomokból nyert értékek.

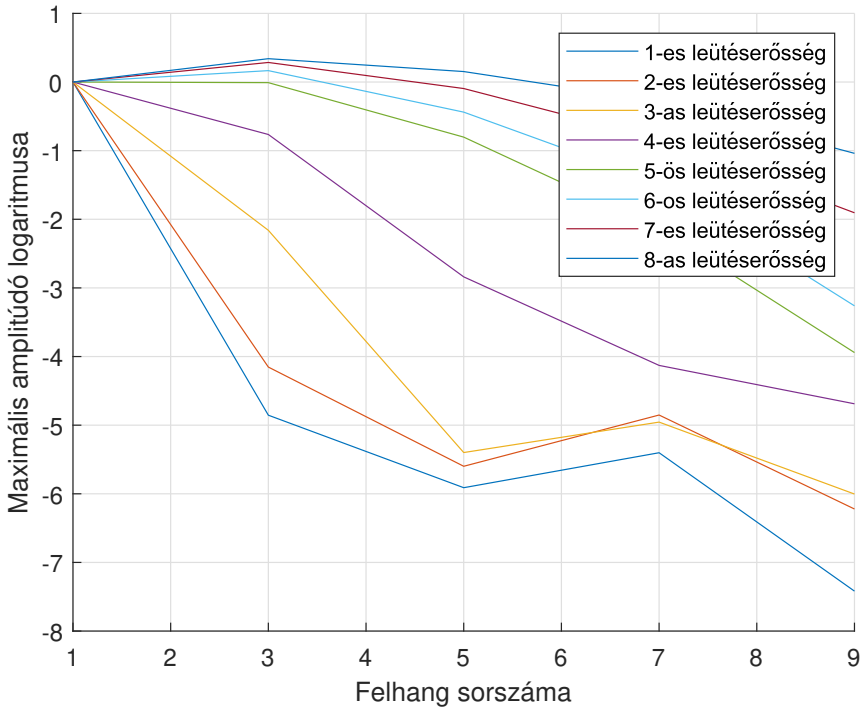
5.1.2. Polinomok illesztése az időállandókra

Az időállandókra való illesztés nagyon hasonló módon történik, azzal a különbséggel, hogy ebben azoknak az adatoknak a helyén, amelyeket nem



11. ábra. A_3 hang felhangjainak időállandói

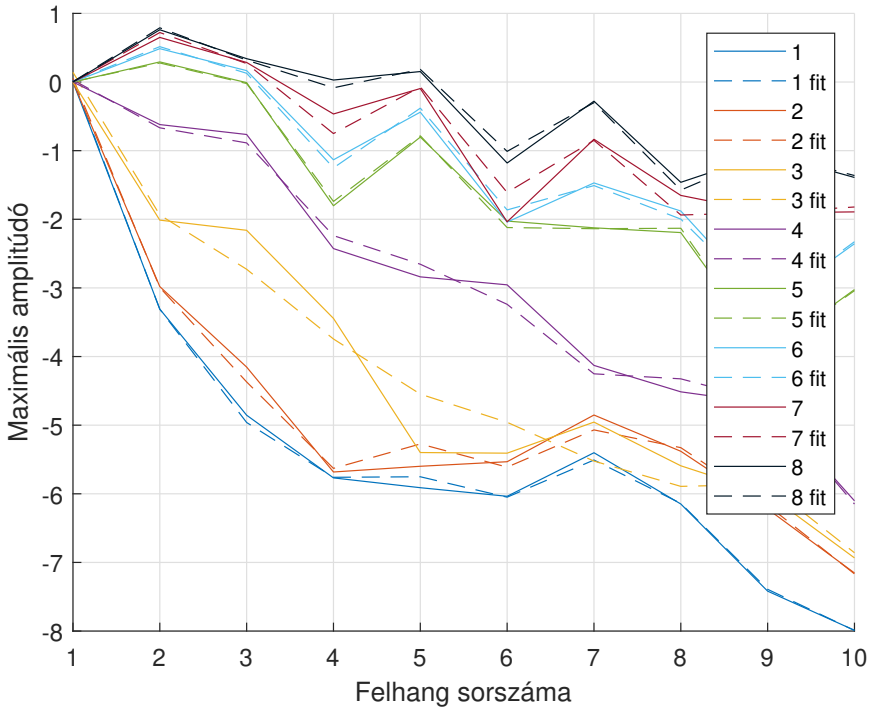
ismerünk, NaN szerepel. Ezeket el kell távolítani a vektorból, mert a *polyfit()* függvény nem tudja ezeket kezelni. Továbbá ebben az esetben nem normáljuk le az adatokat az alapharmonikushoz tartozó értékkel. Az időállandókra elsőfokú polinomot illesztünk, de ebben az esetben is különválasztjuk a páros és páratlan felhangokat. Amit az a 14. ábrán látható, a polinom a magasabb hangok magasabb számú harmonikusoknál nem illeszkedik a legjobban. Ebben az esetben a hozzájuk tartozó amplitúdók értéke nagyon kicsi, ezért ezeknek a hatása nem érzékelhető az így generált hangban. Az alacsonyabb leütéserősségekre viszont tökéletesen illeszkedik a polinom.



12. ábra. A3 hang páratlan felhangjainak maximális amplitúdói, logaritmikus skálán ábrázolva, az alaphang amplitúdóértékével normálva

5.2. További adatok generálása és az adatok szövegfájlba mentése

További adatokat szükséges a maximális amplitúdókra illesztett polinomokat (*poly_A*) és az időállandókra illesztett polinomokat (*poly_tau*) tartalmazó mátrixhoz hozzáadni. Ahhoz, hogy az eredeti Rhodes hangszer teljes hangterjedelmét tudjuk szimulálni, E_1-től E_7-ig, ezekhez a szélső hangokhoz tartozó adatokra is szükségünk van. Mivel ezekről nem állt rendelkezésre hangfelvétel, extrapolálás segítségével határoztuk meg ezeket az amplitúdó- és időállandó-értékeket. Ezek előállítása után az összes köztes hanghoz tartozó érték megkapható interpolálás segítségével.

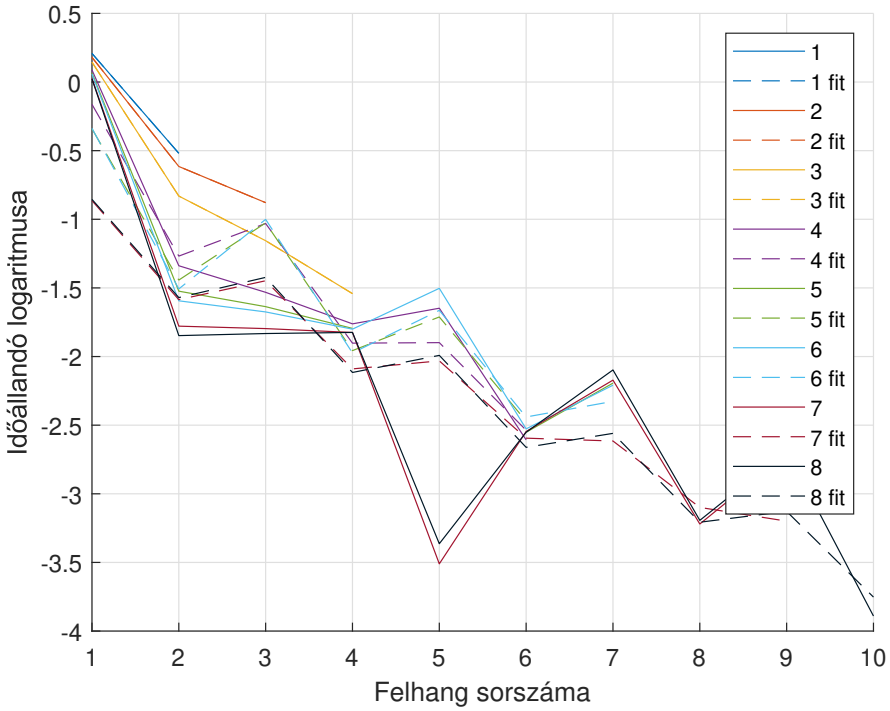


13. ábra. Az A3 hang felhangjainak maximális amplitúdói és a rájuk illesztett polinomok

Amikor már az összes szükséges adat rendelkezésre áll, az adatokat *.txt* fájlokba írjuk. Az amplitúdókat egyenesen a *poly_A.txt* fájlba. Az időállandókat viszont a leütéserősség szerint harmonikusonként átlagoljuk (lásd: 5.2.2. fejezet), és a későbbi interpolálás megkönnyítésére az értékek természetes alapú logaritmusát írjuk a *poly_tau.txt* fájlba.

5.2.1. További amplitúdóadatok generálása

Egyrészt minden hangmagassághoz hozzáadtunk egy 0 leütéserősségű adatot, ebben az esetben az összes felhanghoz tartozó amplitúdó értéke 0. Erre azért van szükség, mert ha egy billentyűt nem ütünk le, akkor nem is fog hangot kiadni a hangszer magából, így megvan az alsó határa a hangok di-



14. ábra. Az A_4 hang felhangjainak időállandói és a rájuk illesztett polinomok

namikájának. Ezt a későbbiekben az interpolálásnál lehet használni.

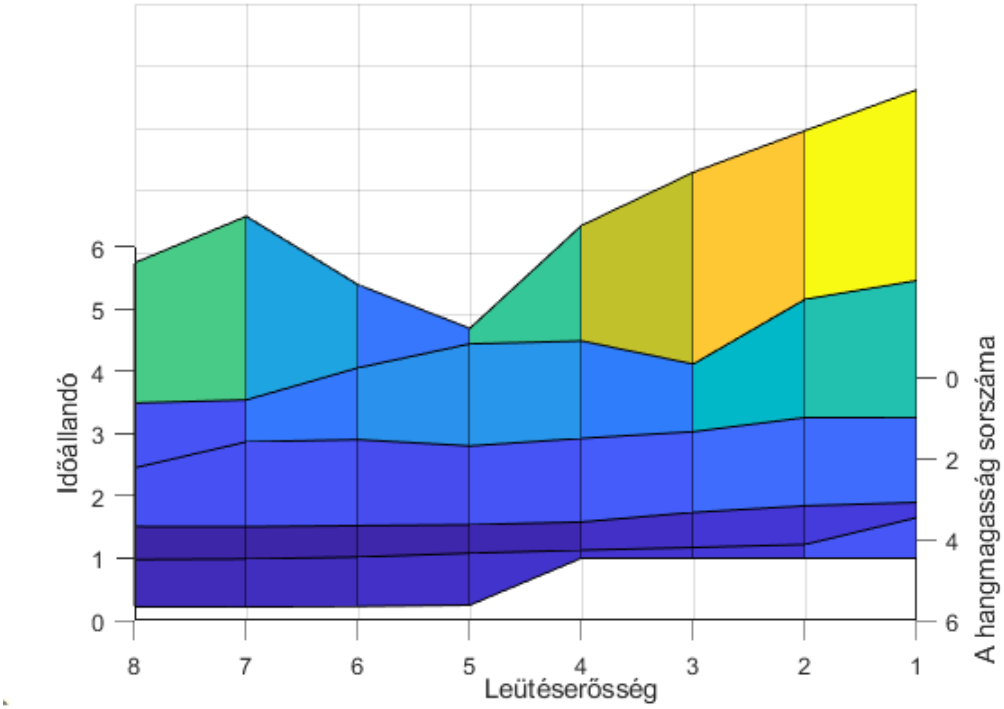
Másrészt pedig a tetszőleges amplitúdóértékek meghatározásához használni kívánt *interp2()* MATLAB függvény, nem tud extrapolálni, így szükség volt arra, hogy a Rhodes legmélyebb (E_1) és legmagasabb (E_7) hangját jellemző amplitúdóértékeket is meghatározzuk. Ez az *interp1()* lineáris extrapoláló funkciójával történt. Mivel ez csak egyetlen dimenzió mentén tud extrapolálni, ezért harmonikusonként külön kell meghatározni a hozzájuk tartozó értékeket. Az extrapolálást az amplitúdóadatok természetes alapú logaritmusán hajtjuk végre, majd alakítjuk vissza az így kapott értéket exponenciális függvény segítségével. Így nem tudnak a valóságnak nem megfelelő negatív maximális amplitúdók kijönni az extrapolált értékekre.

5.2.2. További időállandóadatok generálása

Az 5.3. fejezetben említett *interp1()* megvalósításban nincsen implementálva a MATLAB változatban megtalálható extrapoláló opció. Így az időállandók számításához is szükséges további adatokat generálni a hangterjedelem szélét jelentő hangokhoz. A 17. ábrán megfigyelhető, hogy az időállandók nagysága a leütéserősség változásával nem változik számottevő mértékben. Ezért egy adott hangmagassághoz tartozó, adott sorszámú felhang időállandóját függetlennek tekintjük a leütéserősségtől, és a hozzá tartozó időállandóértékek átlagával közelítjük azt. Ezt a MATLAB *trimmean()* függvényével kapjuk meg, ennek köszönhetően a kiugró értékek eltávolíthatóak az átlagképzésből és így ezek nem módosítják az átlag értékét.

Az adatok extrapolálását szintén az *interp1()* lineáris extrapoláló funkciójával valósítottuk meg. Viszont ez esetben jobb eredményeket kaptunk, ha logaritmikus skálán vett adatokra illesztettük a további pontokat és ebből nyertük ki egy e-adik hatványra emeléssel az időállandók értékeit. Ennek az eljárásnak köszönhetően nem jöhetnek ki irreális negatív időállandók, hiszen az exponenciális függvény használatának következtében csak pozitív időállandóértékek adódhatnak.

Ebben az esetben is felharmonikusonként generáljuk le az adatokat, egy ciklusban az amplitúdóérték extrapolációval, egy adott hangmagasság, egy adott felharmonikusánál viszont itt a fentebb tárgyalt okból ugyanaz az érték kerül minden leütéserősséghez.

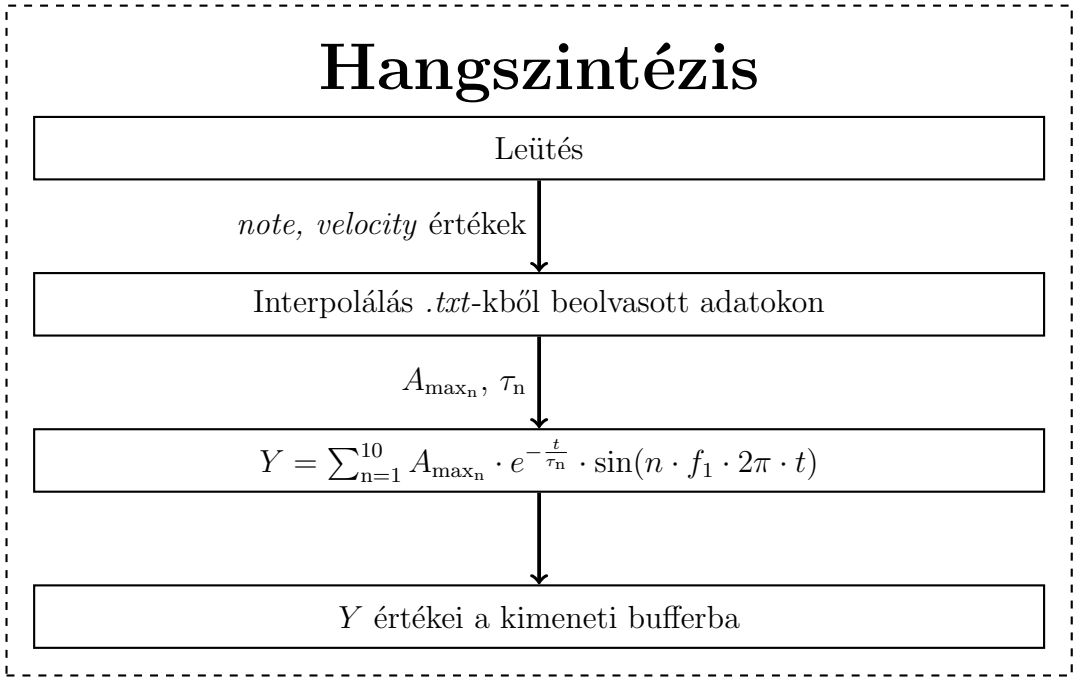


15. ábra. A második felharmonikusokhoz tartozó időállandók, az illesztett polinomok alapján

5.3. Az interpoláló algoritmusok implementálása C++ nyelven

A VST plugin megvalósítása C++ környezetben történt, a technológiát kifejlesztő Steinberg cég által biztosított SDK ebben készült, C API alapra. A plugin használata során egy MIDI billentyű lenyomásából szerzett leütéserősség- és hangmagasság-információhoz kell az előzetesen betöltött adatokon interpolációt elvégezni. Majd ki kell számítani így a hozzájuk tartozó maximális amplitúdó- és időállandó-értékeket minden harmonikusra, hogy ezek segítségével elvégezhető legyen a szintézis.

Ezért szükségünk van a MATLAB *interp1()* és *interp2()* függvényének



16. ábra. A hangszintézis folyamata a VST pluginban

a megvalósítására ebben a környezetben. Az előbbivel az időállandók számítását végezzük, ebben az esetben csak egydimenziós interpolálásra van szükség az 5.2.2. fejezetben tárgyalt egyszerűsítés miatt. Ebben az esetben is az adatok logaritmusára interpolálunk, és ebből számítjuk vissza az időállandókat. Az *interp2()*-re pedig a maximális amplitúdók számításához van szükség, ebben az esetben az interpolációt kétdimenziós felületen kell elvégezni. Mind a két esetben lineáris interpolációt alkalmaztunk, mivel ennek segítségével is jó eredményeket kaptunk, és kisebb a számításigénye, mint a bonyolultabb *Modified Akima*, vagy *spline* interpolációknak.

A függvényeknek Qianlong Lan készített egy C++ megvalósítást, ez elérhető a GitHub-on [8], és én is ezt használtam a program megvalósításához. A C++-ba átírt változat mind az *interp1()*, mind az *interp2()* függvényt megvalósítja. Egyszerre egy, vagy akár több pontra is el tudja végezni az

interpolálást, a jelen esetben csak az előbbire volt szükségünk. A függvények több interpolációs eljárást is kezelni tudnak, mi a lineárisat használtuk belőlük. A legnagyobb hiányossága a megvalósításnak, hogy nem tud extrapolálni, viszont ezt ki lehetett úgy küszöbölni, hogy a szélső adatokat még MATLAB segítségével kiszámítottuk (lásd: 5.2. fejezet).

5.4. A hangminták összeállítása harmonikusonként

A hangminták szintézisét az analízishez hasonlóan harmonikusonként végezzük. A felhasználótól érkező MIDI-jel tartalmazza az adott billentyűhöz tartozó MIDI-hang sorszámát, ez alapján egy táblázatban kikereshető a hozzá tartozó frekvencia. A hangmagasság és a leütéserősség adatok alapján elvégezhető az analízisből nyert adatokon az interpoláció, hogy megkapjuk a felhasználói leütéshez tartozó maximális amplitúdókat és időállandókat minden harmonikushoz. Ez alapján harmonikusonként összeállítható a kívánt hanghoz tartozó hangminta. A hangminták összeállítását a következő képlet szerint végezzük:

$$Y = \sum_{\mathrm{n}=1}^{10} A_{\mathrm{max_n}} \cdot e^{-\frac{t}{\tau_\mathrm{n}}} \cdot \sin(n \cdot f_1 \cdot 2\pi \cdot t), \tag{5.1}$$

ahol:

- Y – a végső hangminta
- n – a harmonikus sorszáma, 10 harmonikust generálunk le a jelen megvalósításban
- $A_{\mathrm{max_n}}$ – az n-edik harmonikushoz tartozó maximális amplitúdó értéke, amit az interpolálás segítségével határoztunk meg
- t – a hangmintához tartozó időtengely

- τ_{n} – az n-edik harmonikushoz tartozó időállandó értéke, amit szintén interpolálás segítségével határoztunk meg
- f_1 – az alaphang frekvenciája, a frekvenciatáblázatból

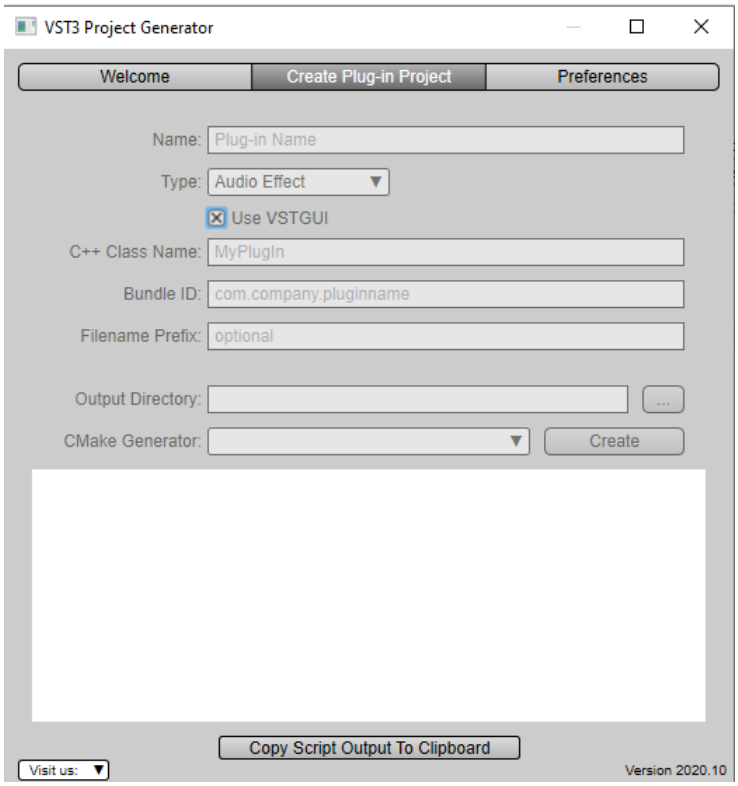
Ezt a modellt használtuk fel a jel analíziséhez is a 4.1. fejezetben.

6. A VST plugin implementálása C++ nyelven

A Steinberg cég lehetővé teszi, hogy külső fejlesztők is létrehozzanak különböző VST pluginokat. A projektek elkészítéséhez rendelkezésre áll a *VST3 Project Generator* program, amely segítségével néhány egyszerű lépésben létre lehet hozni a projekt vázát jelentő fájlokat. Ez után már csak meg kell építeni őket a CMake program segítségével, és így kapunk egy szerkeszthető solution-t.

A generálás után rendelkezésünkre áll a *myplugincontroller.cpp* és a *mypluginprocessor.cpp* forrásfájlok, valamint a hozzájuk tartozó fejlécfájlok, viszont a bennük található függvények megvalósítása már a mi feladatunk. A megvalósítás alapjául a Steinberg honlapjáról elérhető *Note Expression Synth* szoftverhangszer felépítését vettem, ez nagyon sok plusz funkcióval rendelkezik, de az alapvető felépítés is megismerhető belőle.

A generált fájlokon túl el kell készítenünk a *mypluginvoice.cpp* forrásfájlt is, a hozzá tartozó header fájllal egyetemben. Ezek felépítése szintén a *Note Expression Synth*-ben látható fájlon alapul.



17. ábra. A VST3 Project Generator felhasználói felülete

6.1. A program felépítése

- *myplugincontroller* – A plugin vezérléséért felelős részeket gyűjti egybe. Például azt, hogy milyen, a hangot leíró paraméterek legyenek állíthatóak benne.

- *mypluginprocessor* – A plugin működése során jelentkező folyamatokat leíró programrészek találhatóak itt.

- *mypluginvoice* – A megszólalásért felelős rész, itt történik a leütések kezelése, és a hangminták összeállítása is.

6.1.1. Myplugincontroller

Ezekben a programrészekben, a *myplugincontroller.cpp*-ben és a *myplugincontroller.h*-ban, történne a plugin állítható paramétereinek az inicializálása, módosításuknak kezelése. Az én megvalósításomban nincsenek ilyen paraméterek, így ezek az osztályok nagyjából az eredeti, *VST3 Project Generator*-al generált állapotban maradtak, néhány belső változó inicializálását kellett itt elvégezni.

6.1.2. Mypluginprocessor

Az ide tartozó forrás- és header-fájl tartalmazza azokat a kódrészeket, amelyek a program futása közben előforduló különböző eseményeket kezelik, ilyen például a plugin példányosítása. Ebben az esetben ki kellett egészíteni a generált kódokat. A legfontosabb változtatás a *setActive()* és a *process()* függvényekben történt. Az előbbi egy adott leütés esetén, létrehoz a leütéshez egy új *voiceProcessor* megvalósítást, amely a hangminták előállításával foglalkozik. A második pedig meghívja az összes szóló hang *voiceProcessor*-ának a *process()* függvényét, és az így kapott adatot egy kimeneti bufferbe írja.

6.1.3. Mypluginvoice

Itt történik a hangminták előállítása. A program indulásakor itt olvassuk be az analízisből kapott amplitúdó- és időállandóadatokat a hozzájuk tartozó tömbökbe, ezek segítségével interpoláljuk ki a tetszőleges leütéshez tartozó értékeket. Ilyenkor történik meg a MIDI-hangokhoz tartozó frekvenciák meghatározása is.

A hozzá tartozó *process()* függvényben történik a minták előállítása, az 5.4. fejezetben olvasható módszerrel. A DAW-tól kapott bufferméretnek megfelelő mennyiségű adatot a kimeneti bufferbe írjuk.

Van továbbá egy *noteOn()* függvénye is, ez leütéskor hívodik meg. Ebben a rendelkezésre álló hangmagassággal, illetve leütéserősséggel meghívjuk az *interp()* függvényeket, így megkapva a hanghoz tartozó amplitúdó- és időállandó-értékeket. Ezeket tudjuk felhasználni a *process()*-ben ezután a hangminták előállításához. Egy hang hosszát az alapharmonikus időállandójának a 10-szeresében határoztuk meg. Általánosan kijelenthető, hogy az alapharmonikus lecsengése tart a leghosszabb ideig a harmonikusok közül, ennek a 10-szeresénél pedig már elhanyagolható a hangminta amplitúdójának értéke, nem hallunk már pattanást, ha vége van. Egy hang egymás utáni többszöri leütését is tudja kezelni a plugin, ez esetben az előző leütés hangja megszűnik, mint ahogy az történik egy valódi hangszernél, és az új leütés hangja szól csak.

6.2. A plugin tesztelése

A szoftverhangszert a fejlesztés során a *Reaper* DAW szoftverben teszteltem. A *Visual Studio* lehetővé teszi, hogy a debuggolni kívánt programot külső process-ben futassuk, így rendeltetésszerűen tudjuk használni a VST-t, viszont a program nyomon követésére van lehetőség a *Visual Studio*-ban. A DAW-ban a plugint hozzáadtuk egy csatornához, amin MIDI-jeleket kapott, és így lehet ellenőrizni, hogy a hangszer megfelelően működik e.

A tesztelés során több problémába is belefutottam, nem megfelelően definiált időtengely miatt először nem is adott hangot a hangszer, ezt a tengely módosításával sikerült kiküszöbölni. Valamint a felhasználás során találkoztam egy olyan jelenséggel, hogy a leütött hang hangmintái szakadozva voltak hallhatóak. Ez a probléma megoldható a DAW-ban a bufferméret megnövelésével.

A tesztelés során készítettem hangmintákat, ezeket az elektronikus

melléklet tartalmazza.

7. Konklúzió

A dolgozat célja egy olyan szoftverhangszer megalkotása volt, amely a felhasználó számára az eredeti hangszeréhez nagyon közeli hangzásélményt nyújt. Ennek elérésére nagy mennyiségű, jó minőségű felvételt készítettem a mintául szolgáló hangszer teljes hangterjedelmén elhelyezkedő 6 hangról. Ezek elemzését nagy pontossággal végeztem el MATLAB környezetben. A modellillesztés során egyszerűsítésekkel éltem annak érdekében, hogy a szintézis folyamatát egyszerűbbé tegyem, így annak számításigényét csökkentsem. Majd ez alapján elkészítettem a VST plugint, amely bár még nem rendelkezik felhasználói felülettel, a kitűzött célnak megfelelően az általam rögzített Rhodes-hoz nagyon hasonló hangja van. Így elkészült az a szoftverhangszer, amelya Rhodes zongora hangzásvilágát adja vissza, és egyszerű MIDI-billentyűzettel működtethető.

7.1. Továbbfejlesztési lehetőségek

Bár nagy mennyiségű hangmintával dolgoztunk, az interpolálás helyes működéséhez a két szélső hanghoz tartozó értékeket extrapolálás segítségével kaptuk meg. A valósághoz közelebbi eredményeket kaphatnánk valószínűleg akkor, ha ezekről a hangokról is készítenénk felvételeket. És a hozzájuk tartozó adatokat a többi mintánál is alkalmazott hanganalízis segítségével kapnánk.

A jelenleg alkalmazott modell elhanyagolja a nem harmonikus jellegű hangokat. Ez a nagy leütéserősségeknél, valamint az alacsony hangok gyenge leütésénél, ahol ezek leginkább jelentkeznek hallható is a hiányuk. Ezen jelenségek alaposabb tanulmányozásával a hangszer hangzása valósághűbbé tehető.

A szoftverhangszer használata felhasználóbarátabbá tehető még a Graphical User Interface (GUI) továbbfejlesztésével, jelen formájában nem állíthatóak rajta az alapvető paraméterek sem, mint pélsául a hangerő. További paraméterek, például az eredeti hangszeren is megtalálható *Bass Boost* hozzáadása is elősegítené, hogy a felhasználó az általa elképzelthez közelebb álló hangot tudjon kinyerni a VST-ből.

A hangszer használatát továbbá meg lehetne könnyíteni azzal, hogy a hangszerhez rendszeresen használt erősítőt, esetleg a 1.3. fejezetben említett gyakran alkalmazott effekteket is beépítsük a szoftverhangszerbe. Így egy plugin-ban lenne elérhető az egész általánosan használt hangszer, effektek, erősítő összeállítás, nem kéne ezt külön plugin-okból összeállítani, így kompaktabbá és egyszerűbbé téve egy viszonylag elterjedt használati módot.

A játékélmény még növelhető lenne azzal, ha a programmal lehetne használni *sustain pedal*-t, hiszen ez egy nagyon elterjedt kiegészítő a különféle billentyűs hangszereknél.

7.2. Köszönetnyilvánítás

Nagyon köszönöm a konzulensemnek, Dr. Rucz Péternek az elmúlt egy évben nyújtott kiapadhatatlan segítségét, amely nélkül ez a dolgozat nem készülhetett volna el. Köszönöm Beke Istvánnak a felvételekhez használt hangszert. Valamint köszönöm édesanyának a dolgozat többszöri átolvasását.

Hivatkozások

[1] *Rhodes Manual*. URL: https://www.fenderrhodes.com/org/manual/ch4.html#4-7 (elérés dátuma 2022. 11. 30.).

[2] URL: https://www-origin.steinberg.net/technology/ (elérés dátuma 2022. 11. 30.).

[3] *MIDI History:Chapter 6-MIDI Is Born 1980-1983*. URL: https://www.midi.org/articles/midi-history-chapter-6-midi-is-born-1980-1983 (elérés dátuma 2022. 11. 30.).

[4] John Ellinger. *MIDI Basics*. 2014. URL: https://people.carleton.edu/~jellinge/m208w14/pdf/02MIDIBasics_doc.pdf (elérés dátuma 2022. 11. 30.).

[5] Tamara Smyth. "Signal Analysis". Dipl. Department of Music, University of California, San Diego, 2019. URL: http://musicweb.ucsd.edu/~trsmyth/analysis/ (elérés dátuma 2022. 11. 30.).

[6] Fiala Péter és Rucz Péter. *Hangjelek digitális feldolgozása az akusztikai gyakorlatban*. Mérési segédlet. URL: https://last.hit.bme.hu/sites/default/files/documents/fft_leiras.pdf.

[7] Nemes Dániel. "Fender Rhodes elektromos zongora modellezése MATLAB és VST környezetben". Szakdolgozat. Budapesti Műszaki és Gazdaságtudományi Egyetem, 2020.

[8] Qianlong Lan. *Interp2 C++ implementation*. 2020. máj. URL: https://github.com/lanqianlong/Interp2CPP (elérés dátuma 2022. 11. 30.).