Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Méréstechnika és Információs Rendszerek Tanszék

Szabó Gergő

FIZIKAI ALAPÚ GITÁRHANGSZINTÉZIS

KONZULENS

Dr. Bank Balázs

BUDAPEST, 2013

FELADATKIÍRÁS

Szabó Gergő László (DFRRZE)
szigorló villamosmérnök hallgató részére

Fizikai alapú gitárhangszintézis

A fizikai alapú hangszintézis alapgondolata, hogy a hangszer egyes részeinek rezgését modellezi, így a hagyományos módszerekkel nehezen figyelembe vehető hatások (pl. húrok csatolt rezgése) könnyebben szimulálhatók. A módszer további előnye, hogy a hangzás a zenészek számára könnyen értelmezhető valós fizikai paraméterekkel (pl. a húr tömege, pengetés sebessége) befolyásolható.

A húros hangszerek fizikai alapú modellezése akadémiai szinten kiforrottnak tekinthető, ugyanakkor az ezt alkalmazó termékek csak az utóbbi években jelentek meg. A számítógépek sebességének növekedése lehetővé teszi, hogy a fizikai alapú szintetizátort célhardver alkalmazása nélkül, szoftveres környezetben valósítsuk meg. Ez leggyakrabban a Steinberg cég VST környezetében történik, melynek előnye, hogy így az elkészített plugin szinte bármely zeneszerkesztő programban használható. Emellett a pluginként történő megvalósítás leveszi a fejlesztő válláról a hangkártyakezelés nehézségeit. A hallgató feladata egy VST alapú gitárszintetizátor megvalósítása.

A hallgató munkájának a következőkre kell kiterjednie:

- Tekintse át a különböző húrmodellezési eljárások irodalmát!
- Valósítson meg C++ nyelven egy VST alapú gitárhang-szintetizátort a húr modális modellje alapján!
- A húrmodell a húr mindkét polarizációjának rezgését vegye figyelembe, a modell paramétereit felvett gitárhang analízise alapján állítsa be!
- A pengetés modellezésénél fizikai alapú gerjesztésmodellt alkalmazzon!
- A gitártest hatását akusztikus gitáron végzett átviteli függvények mérése alapján tervezett digitális szűrővel modellezze!
- A megvalósított VST plugin adjon lehetőséget a hangszer főbb paramétereinek megváltoztatására!

Tanszéki konzulens: Dr. Bank Balázs, docens

Budapest, 2013. október 10.

……

Dr. Jobbágy Ákos

tanszékvezető

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Szabó Gergő**, 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, 2013. 12. 19.

…….

Szabó Gergő

Összefoglaló

Az elmúlt néhány évtizedben népszerűvé vált zeneszerkesztő szoftverek és digitális szintetizátorok a kívánt hangszer hangját külső vagy beépített úgynevezett pluginek (bővítmények) használatával állítják elő. E dolgozat témája egy ilyen szintetizátor plugin készítése, amely a klasszikus gitár hangját reprodukálja.

Húros hangszerek szintézisére több módszer is elterjedt. Ezek egyike a fizikai alapú szintézis, amely a hangszer egyes jól elkülöníthető részeit (húr, hangszertest, zenész) külön-külön blokkokban valósítja meg, és az ezek között fellépő kölcsönhatásokat fizikai egyenletekkel írja le. A módszerrel nagy pontossággal lehet modellezni bármilyen húros hangszert, viszont a fizikai leírást finomítva a számítási igény is növekszik, ami egy szintetizátornál kulcsfontosságú probléma. Emiatt egészen a kilencvenes évekig nem használták valós idejű hangszintézisre ezt a megoldást. Mára azonban a számítástechnika fejlődésével a középkategóriás számítógépekben található mikroprocesszorok is képesek fizikai alapú hangszermodellek futtatására.

A húr modellezésére is többféle módszer létezik. Én a modális szintézist választottam, amellyel igen jól lehet közelíteni a valódi hangszer tulajdonságait. Ez a módszer a hullámegyenlet frekvenciatartománybeli megoldásán alapul, a húr rezgéseit módusokra bontja fel. A paraméterek megállapítása mérésekből történt.

A megvalósított modellben kétféle gerjesztési mód szerepel: a húrokat pengetővel és ujjal is lehet pengetni. A szintetizátor használatakor a felhasználó választhat a két pengetési mód között. A hangszertestet párhuzamos másodfokú szűrők modellezik, amelyek paramétereit a gitártest méréséből és analíziséből állapítottam meg.

A virtuális szintetizátort VST (Virtual Studio Technology) környezetben implementáltam annak érdekében, hogy zeneszerkesztő szoftverekkel kompatibilis legyen. A dolgozat végterméke egy olyan, *.dll* kiterjesztésű VST plugin, amelyet erre alkalmas, úgynevezett digitális audió munkaállomás szoftverekben mint szintetizátor lehet használni, illetve a plugint számítógépre kötött MIDI-billentyűzet segítségével valós időben, közvetlenül is lehet vezérelni.

Abstract

In the last few decades digital synthesizers and music producing software have became increasingly more popular. These software products create the sound of the desired instrument using external or built-in plugins. The goal of this thesis is to create a plugin that reproduces the sound of the classical guitar.

There are several methods for synthesizing string instrument sounds, one of which is called physical modeling synthesis. This method models the elements of an instrument (the strings, the instrument body, the musician) as separate blocks. The interaction between these blocks is described by physical equations. Using this method any string instrument can be modeled with good accuracy, but the more precise the model, the more the computational cost, which is crucial in audio applications. Physical modeling synthesis wasn't used for real-time sound synthesis because of this fact until the 1990s. Nowadays however the computational capacity of the average personal computer is sufficient for running a physical instrument model.

For modeling the string there are several methods available. In this thesis modal synthesis was chosen, because it can replicate a real instrument with great accuracy. The model parameters are obtained by recording, measuring and analyzing the sound of a classical guitar.

There are two ways for exciting the string in the final instrument model: it can be plucked using a plectrum (pick) model or with fingers. When using the VST plugin, the user can choose from the two excitation models. The instrument body is modeled with parallel second-order filters, whose parameters have been obtained from the analysis of measured guitar body response.

The virtual instrument was implemented in Steinberg's VST (Virtual Studio Technology) environment. The final result of this work is a *.dll* file, which can be imported and used as a virtual synthesizer in DAW software. The plugin can also be controlled directly in real time with a MIDI-keyboard.

1 Bevezetés

A klasszikus gitár a 19. század óta egészen napjainkig töretlen népszerűségnek örvend, mind a professzionális, mind a hobbizenészek körében. Köszönheti ezt annak a tulajdonságának, hogy viszonylag kevés energiabefektetéssel sikerélményeket lehet vele elérni, valamint könnyű vele többszólamú dallamjátékot megvalósítani.

A klasszikus gitárt műanyaghúrokkal szerelik, többnyire az akusztikus (western) gitárnál szélesebb fogólap és rövidebb nyak jellemző rá, de ez nem követelmény. Zenei stílusok közül a klasszikus zene, spanyol flamenco, bossa nova, a jazz és a reggae bizonyos irányzataiban használják főképp. Stílustól függően pengetővel vagy ujjal is szoktak rajta játszani. Pengetővel hangosabb, karakteresebb hangzást lehet elérni, valamint dinamikusabb játékot, ujjal pedig többszólamú dallamjátékot könnyebb megvalósítani, amelyre például a klasszikus zenében van igény.

Az elmúlt néhány évtizedben egyre nagyobb népszerűségre tettek szert azok a művészek, akik hangszeres játék helyett digitálisan állítják elő a zeneszámaikat. Ehhez régen stúdiókban, hardveres szintetizátorokkal dolgoztak, manapság a virtuális stúdiók, digitális audio munkaállomások (DAW – Digital Audio Workstation) a jellemzőek. Utóbbiak olyan szoftverek, amelyekkel egy megírt dallamot virtuális szintetizátorokkal lehet megzenésíteni, meglévő hangot effektezni, keverni, módosítani. A legjobb ilyen virtuális szintetizátorok olyan valósághűen tudják utánozni az adott hangszer hangját, hogy a hallgató számára sok esetben nem egyértelmű, hogy valódi, vagy virtuális hangszert hall.

Ezek után érthető, hogy igény van egy olyan virtuális szintetizátorra, amely élethűen adja vissza egy klasszikus gitár hangját, és kellően rugalmas, hogy a hangzás testreszabható legyen. Az interneten fellelhető ingyenes gitárszintetizátorok többsége azonban nélkülöz mindenféle fizikai alapot vagy mérést, és sokszor a minőségük is hagy kivetnivalót maga után.

E szakdolgozat célja egy olyan virtuális klasszikusgitár-szintetizátor megvalósítása, amely elméleti alapokon nyugszik, fizikai egyenletekből és mérésekből származnak a paraméterei. A fizikai alapú hangszintézis lényege, hogy a hangszer egyes jól elkülöníthető részeit (húrok, hangszertest), valamint a játékos hatását (pengetőt vagy

ujjat) külön modellezzük, és a köztük lévő kölcsönhatásokat fizikai egyenletek alapján számítjuk. Ez számításigényesebb, mint más, korábban használt megoldások, de a számítástechnika fejlődésével mára eljutottunk arra a szintre, hogy egy átlagos számítógép is megbirkózik egy fizikai alapú virtuális szintetizátor futtatásával, több példányban is.

A szakdolgozat elsőként bemutatja a manapság használt hangszintézis-módszereket, megvizsgálva azok előnyeit és hátrányait a gitárszintetizátorban való alkalmazhatóság szempontjából. Külön fejezetet kapott a megvalósított fizikai alapú szintézis: rövid bevezető után a virtuális hangszer részeinek bemutatása következik. Az analízisről szóló fejezetben a gitár és a húrok paramétereinek méréséről és azok feldolgozásáról lesz szó. Végül pedig a VST szabvány és az abban történő implementáció leírása következik.

2 Hangszintézis-módszerek

A szoftveres (virtuális) vagy hardveres (digitális hangszer) szintetizátorok többféle módszerrel állítják elő a kívánt hangot. Ezek között mind bonyolultságban és számításigényben, mind hangzásban vannak különbségek, és a konkrét megvalósítástól függően a végeredmény minősége széles határok között változik.

Hangszintézis alatt természetesen nem feltétlenül létező hangszer imitálását értjük. Főképp elektronikus zenében, de más stílusokban is fontos szerepet játszanak a „géphangok”: olyan szintetizált hangok, amelyek előállításánál nem cél, hogy hasonlítsanak bármi más létező hangra.

Az alábbi megközelítés Smith [1991] és Bank [2000] csoportosítására épít.

2.1 Absztrakt algoritmusok

Az absztrakt algoritmusok matematikai műveletek segítségével állítanak elő hangot. A módszer jellegéből adódóan valódi hangszert nagyon bonyolult vele modellezni, viszont új, szintetikus hangok előállítására kiválóan alkalmazható.

A frekvenciamodulációs hangszintézist [Chowning 1973] ismerteti. Maga a módszer a rádiózásban használt frekvenciamoduláción alapul, azzal a különbséggel, hogy itt a „vivőfrekvencia” és a „modulálófrekvencia” összemérhető. A használt alapegyenlet [Chowning 1973] szerint:

$$e = Asin(\alpha t + Isin(\beta t))$$

ahol

e = modulált jel pillanatnyi amplitúdója,
α = vivőfrekvencia [rad/s],
β = modulálófrekvencia [rad/s],
I = modulációs index, amely a modulálójel maximális kitérésének és a modulálófrekvenciának aránya.

A módszer előnye, hogy már egy kétoszcillátoros rendszerrel is sokféle jel állítható elő, valamint inharmonikus hangzás is generálható. Szinuszos modulálójel és vivőjel esetén a felharmonikusok a Bessel-függvények segítségével számíthatók.

A waveshaping szintézis egy egyszerű bemenőjel nemlineáris torzításán alapul ([Le Brun 1979] és [Arfib 1979]). Előnye, hogy a bemenőjel amplitúdóváltozása nagyban változtatja a kimeneti spektrumot. Szinuszos jel esetén a kapott harmonikusok amplitúdóját a Csebisev-polinomok adják. Így a formálójel analitikusan meghatározható a felhangok amplitúdóiból. Ezzel a módszerrel csak harmonikus spektrumot lehet generálni.

2.2 Tárolt minták visszajátszása

E módszer alapja a kívánt hangok előre rögzítése, majd az adott billentyűk lenyomására azok visszajátszása. Ebből adódóan azt a bizonyos hangot nagy pontossággal adja vissza, viszont a hangszerből és a zenész játékából adódó paraméterváltozásokat nem tudja valósághűen követni. Maga a visszajátszási algoritmus nagyon egyszerű, kevéssé számításigényes, de a minták tárolása esetenként sok memóriát igényel.

A kereskedelemben kapható szintetizátorok és digitális zongorák döntő többsége ezt az algoritmust használja. Minden hangot hangszerenként külön tárolnak a memóriában, és az adott billentyű megnyomására visszajátsszák [Roads 1995]. A memóriaigényt csökkenti, hogyha a hang állandó részét egy rövid mintából ismételgetve állítják elő. A játékstílusból adódó hatásokat burkológörbékkel és szűrőkkel modellezik.

Mivel ez a módszer a hangokat külön-külön kezeli, a hangszerek csatolt rezgéseit nem tudja szimulálni. Hátránya továbbá az is, hogy azon hangszereknél, ahol a zenésznek közvetlen befolyása van a húrra (vagy egyéb rezgőtestre), ott ezeket a hatásokat nem lehet valós időben szimulálni.

Ettől függetlenül a tárolt hangmintákon alapuló szintézis igen hatékony lehet, egyszerűsége miatt elterjedt, és például zongora modellezésénél meghatározó fontosságú, mivel azt megfelelő minőségben lehet előállítani.

2.3 Spektrális szintézis

Ezek a módszerek időtartomány helyett frekvenciatartomány felől vizsgálják az előállítandó hangot. Ezzel az emberi hallásra jellemző tulajdonságokat is be lehet építeni a modellbe, a frekvenciatartománybeli elfedési jelenségeket figyelembe lehet venni (például egy domináns harmonikus a kicsivel előtte és utána álló harmonikusokat

elnyomja, nem halljuk őket). A módszer hátránya, hogy a tranzienseket nehézkes vele modellezni, és általában egy hangszer leírásához sok paraméterre van szükség.

A legegyszerűbb változat az additív szintézis, amelyben különböző frekvenciájú és lecsengési idejű, esetleg eltérő fázisú lecsengő szinuszokat összegeznek. Nagy előnye, hogy már meglévő jelből viszonylag egyszerűen kinyerhetők a paraméterek Fourier-transzformáció segítségével, illetve ezek a paraméterek rugalmasan állíthatók. Hátránya, hogy nagy számú paramétert kell tárolni hozzá.

Serra és Smith [1990] szétbontja a jelet a frekvenciatartományban determinisztikus és sztochasztikus komponensekre. Verma és Meng [1995] különveszi a tranzienseket is.

A spektrális szintézis a hangszer által kiadott hangot reprodukálja. Mint ilyen, korlátozottak a lehetőségek a vezérlésre, és egy spektrális szintézisen alapuló gitármodellben a pengetés hatásának modellezése nehézkes, ezért nem feltétlenül az ideális választás egy rugalmas, vezérelhető gitárszintetizátor megvalósítására.

2.4 Fizikai alapú szintézis

Az előzőekben bemutatott módszerek a hangszer által előállított jelet modellezték, akár idő-, akár frekvenciatartományban vizsgálva azt.

A fizikai alapú szintézis ehelyett a hangszert magát modellezi, és a jel helyett a hangforrást írja le fizikai összefüggések alapján. Ezzel elméletben a legvalósághűbb hangzást tudná elérni, hiszen a modell paraméterei valós fizikai paraméterek, és a zenész játékát a bemenetekkel lehet szimulálni. Azonban a paraméterek meghatározása, mint az látni fogjuk, sokszor nehézkes, és a számítási igény is lényegesen nagyobb, mint az előző esetekben.

Rugalmassága és magas szintű testreszabhatósága miatt én ezt a módszert használtam, így a következő fejezetben fejtem ki részletesen.

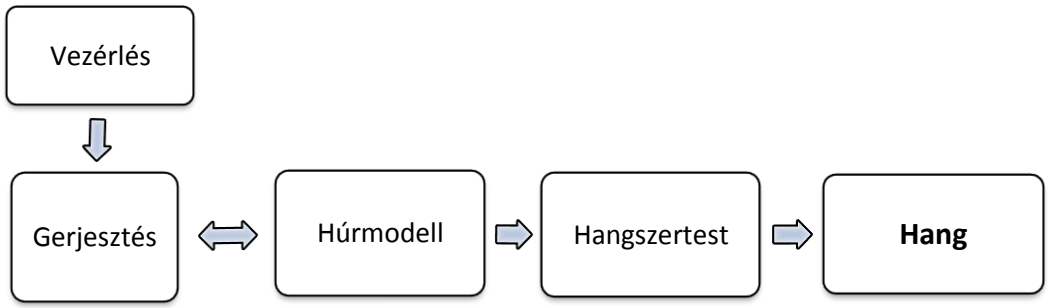
3 A fizikai alapú hangszintézis

A fizikai alapú hangszintézisben a hangszer egyes részeit modellezzük, majd e rendszerek között teremtünk kapcsolatot. Az egyes blokkokat fizikai egyenletek, differenciálegyenletek írják le, amelyeket diszkretizálva áll elő a modell.

Azon túl, hogy a digitális hangszer hangzásának változtatásában másféle megközelítést jelent, a fizikai alapú hangszintézis arra is megoldást nyújthat, ha hangszerkészítő mesterek szeretnének új hangszereket, felépítés-, illetve anyagbeli változtatásokat szimulálni, tesztelni. Természetesen ehhez teljesen másra kell helyezni a hangsúlyt, és a fizikai hatások, egyenletek is sokkal bonyolultabbak, részletesebbek. Ebben a szakdolgozatban nem ez a cél, hanem egy minél inkább valósághű hangzású VST plugin elkészítése, amelyet optimalizálni is kell, mert a processzor számítási kapacitása szűk keresztmetszet lehet. Ennek érdekében bizonyos elhanyagolásokat, egyszerűsítéseket végeztünk a teljes fizikai modellen.

3.1 A modell

Bank [2010] jól összefoglalja egy fizikai alapú zongoraszintetizátor részeit. A modell a gitárra is hasonlóképpen néz ki, a húrok longitudinális rezgését elhanyagolva (3.1-1. ábra):



3.1-1. ábra: húrmodell blokkvázlata

A modell központi eleme a húrmodell, amely kölcsönhatásban áll a gerjesztéssel. A vezérlés blokk tartalmazza azokat a paramétereket, amelyeket a felhasználó közvetlenül állíthat, vezérelhet, és nem a hangszerre jellemzőek, hanem a

zenészre. Az így előálló gerjesztésmodell (ami végeredményben csak a húrmodell kezdeti értékeit állítja be, mint később látni fogjuk) kölcsönhatásban áll a húrmodellel, amelyet magára hagyva a rendszer lecsengő rezgést végez. A húrlábon (angolul bridge) keresztül ez a rezgés átadódik a gitártestnek, amelyet egy szűrővel modellezünk. Az így megszűrt jel pedig maga a kimenet.

3.2 Húrmodell

A húrmodell alapja a húron terjedő hullám egyenlete, amely merev, veszteséges húrra a következőképpen néz ki, Morse és Ingard [1968] szerint, [Bank 2000] alapján:

$$\mu \frac{\partial^2 y}{\partial t^2} = T_0 \frac{\partial^2 y}{\partial x^2} - ES\kappa^2 \frac{\partial^4 y}{\partial x^4} - 2R\mu \frac{\partial y}{\partial t} + d_y(x,t), \qquad (3.2.1)$$

ahol:

- T_0 a húr megfeszítettsége
- μ a húr egységnyi hosszra jutó tömege
- $ES\kappa^2$ a húr merevségét fejezi ki (ezt a gitármodellben elhanyagoljuk)
- R a súrlódási ellenállás, ez a tag a húr veszteségét fejezi ki
- $d_y(x,t)$ a hosszegységre jutó külső erő.

Ez a Helmholtz-egyenlet, kiegészítve a húr merevségére és a veszteségekre vonatkozó tagokkal. A veszteségeket és a merevséget, valamint a külső gerjesztést elhagyva az ideális húr egyenlete:

$$\mu \frac{\partial^2 y}{\partial t^2} = T_0 \frac{\partial^2 y}{\partial x^2} \qquad (3.2.2)$$

A transzverzális hullám haladási sebességet felírhatjuk:

$$c_t = \sqrt{\frac{T_0}{\mu}} \qquad (3.2.3)$$

(3.2.2) és (3.2.3) egyesítéséből pedig:

$$\frac{\partial^2 y}{\partial t^2} = c_t^2 \frac{\partial^2 y}{\partial x^2} \qquad (3.2.4)$$

A differenciálegyenlet megoldására és diszkretizálására három módszer elterjedt: a digitális waveguide-módszer, a véges differenciák módszere, és a modális

módszer. A következőkben röviden bemutatom az első kettőt, majd a harmadikat részletesen, mivel ezt implementáltam a megoldásomban.

3.2.1 Véges differenciák

A véges differenciák módszerét Hiller és Ruiz [1971a,b] ismerteti, ahol az első fizikai alapú szintetizátor alapjaként szolgál. A módszer előnye, hogy közvetlenül a hullámegyenletből jön, illetve kézenfekvő a használata két-, és háromdimenziós struktúráknál [Bank 2006]. Hátránya a magas számításigény, és a diszkretizáció miatt a módusfrekvenciák torzulást szenvednek (diszperzió). Ezzel a módszerrel nincs szükség a hullámegyenletek analitikus megoldására.

Ez a megközelítés a (3.2.4) parciális differenciálegyenlet megoldására a parciális deriváltakat véges differenciákkal helyettesíti. Ideális húrra, ha a felbontás $x_m = m\Delta x, t_n = n\Delta t$, akkor

$$\left.\frac{\partial^2 y}{\partial x^2}\right|_{x_m,t_n} \approx \frac{y_{m-1,n}-2y_{m,n}+y_{m+1,n}}{\Delta x^2} \qquad (3.2.5a)$$

$$\left.\frac{\partial^2 y}{\partial t^2}\right|_{x_m,t_n} \approx \frac{y_{m,n-1}-2y_{m,n}+y_{m,n+1}}{\Delta t^2}, \qquad (3.2.5b)$$

ahol $y_{m,n} = y(x_m, t_n)$. (3.2.5)-öt visszahelyettesítve (3.2.4)-be:

$$y_{m,n+1} = \frac{c_t^2 \Delta t^2}{\Delta x^2}\left(y_{m-1,n} - 2y_{m,n} + y_{m+1,n}\right) - y_{m,n-1} + 2y_{m,n} \qquad (3.2.6)$$

Ez az egyenlet megadja az *m* pont helyzetét a következő időpillanatra, *m-1*, *m+1*, és *m* pont korábbi értékeiből.

Ha úgy választjuk meg Δx és Δt értékét, hogy egy időegység alatt pontosan egy távolságegységet halad a hullám, azaz $c_t = \frac{\Delta x}{\Delta t}$, amint Hiller és Ruiz [1971a] javasolja, akkor

$$y_{m,n+1} = y_{m-1,n} + y_{m+1,n} - y_{m,n-1} \qquad (3.2.7)$$

Ebben az esetben nincsen diszperzió. Belátható, hogy ha a húrt *m* pontban gerjesztjük egységimpulzussal, két egységimpulzus indul el jobbra és balra, és a formájuk megtartásával haladnak. Így azonban elvész a modell rugalmassága. Mivel Δx, Δt és c_t nem függetlenek, a húr alapfrekvenciája csak a Δx húrszakaszok számának (*M*) változtatásával hangolható, ami szükségtelenül nagy *M*-hez vezet a mély húrok esetében.

A veszteségek modellezésénél további problémákba ütközünk. Mivel az R veszteségi ellenállás értéke frekvenciafüggő, a veszteségi tagot nem lehet közvetlenül véges differenciák módszerével diszkretizálni. Hiller és Ruiz [1971a], valamint Chaigne és Askenfelt [1994] megoldást nyújtanak erre a problémára, de mivel nem ezt a módszert valósítottam meg, a hosszadalmas levezetést nem ismertetem.

3.2.2 Digitális hullámvezető

A digitális waveguide modellt Smith [1983, 1992] mutatta be. Ez a megoldás a legelterjedtebb, ugyanis az egydimenziós egyenlet időtartománybeli megoldása igen hatékonyan implementálható digitális jelfeldolgozó eszközök segítségével. A rendszer egy késleltetéssorozatból és a visszacsatoló ágban egy szűrőből áll, amit egyszerű megvalósítani, mégis megtartja a modell fizikai jellegét. Smith [2005] jól összefoglalja a digitális waveguide módszert.

A modell alapja a hullámegyenlet haladóhullám-megoldásának mintavételezése mind az időt, mind a pozíciót tekintve. Maga a haladóhullám-megoldás:

$$y(x,t) = y^{+}\left(t - \frac{x}{c_t}\right) + y^{-}\left(t + \frac{x}{c_t}\right), \qquad (3.2.8)$$

ahol y^{+} és y^{-} az előre-, és hátrafelé (jobbra és balra) haladó hullámok, c_t pedig a transzverzális hullámterjedési sebesség. Ha ezt diszkretizáljuk úgy, hogy $x_m = m\Delta x$ és $t_n = n\Delta t$, akkor:

$$y(x_m, t_n) = y^{+}\left(n\Delta t - \frac{m\Delta x}{c_t}\right) + y^{-}\left(n\Delta t + \frac{m\Delta x}{c_t}\right) \qquad (3.2.9)$$

Ez egyszerűen implementálható a két hullám mintáinak eltárolásával, majd minden időlépésnél az egyes vektorok tartalmának jobbra vagy balra shiftelésével.

A digitális waveguide modellben $\Delta x = c_t \Delta t$, amelyet feltettünk (3.2.7)-hez is. Ezzel az egyszerűsítéssel (3.2.9) a következő alakra kerül:

$$y_{m,n} = y^{+}_{m-n} + y^{-}_{m-n}\,, \qquad (3.2.10)$$

ahol $y^{+}_{m-n} = y^{+}(\Delta t(m-n))$ és $y^{-}_{m-n} = y^{+}(\Delta t(m+n))$. Ha az aktuális időpillanatbeli hullámformákat $y^{+}_{m,n}$ és $y^{-}_{m,n}$ vektorok tárolják, a hullámterjedés modellezéséhez minden időlépéskor egy pozícióegységnyit el kell shiftelni a vektorok tartalmát:

$$y^{+}_{m,n+1} = y^{+}_{m-1,n} \qquad (3.2.11a)$$

$$y^{-}_{m,n+1} = y^{-}_{m+1,n} \qquad (3.2.11b)$$

Veszteséges húrnál a haladó hullámok nem tartják meg a formájukat, hanem időben lecsengenek. A frekvenciafüggetlen veszteségeket erősítések formájában be lehet szúrni a shiftelések (késleltetések) közé. Frekvenciafüggő veszteségek modellezéséhez digitális szűrőket kell beszúrni, azonban ez nem hatékony a digitális szűrők számításigénye miatt.

Megoldást nyújt erre a problémára az, hogyha minden egyes húrszakasz helyett csak kitüntetett pontokban foglalkozunk a veszteségekkel. Ezt megtehetjük, mert csak a húrmodell gerjesztése és kimenete fontos számunkra, a közbülső műveletek nem. Mivel a rendszer lineáris és időinvariáns, a szűrőket bárhova elhelyezhetjük a késleltetések közé, ezért ésszerű választás egy darab szűrőben implementálni az összeset. Ezzel a számítási igény jelentősen javul, így alapvetően ez a megoldás az elterjedt. Az egy szűrőt tartalmazó modell hátránya viszont azt, hogy a szűrő fokszámától függően csak közelíteni tudja a modellezendő húrt, emiatt jelentős hátrányba kerül például a sokkal rugalmasabb modális húrmodellhez képest.

Végeredményben nem ezt a módszert választottam, így további részletezését mellőzöm. Részletes leírást tartalmaz a paraméterbecslésről és a szűrőtervezésről [Bank 2006].

3.2.3 Modális szintézis

A modális szintézis alapötlete, hogy egy rezgő rendszer szétbontható lecsengő egymódusú rendszerekké. Ezzel hasonlít a spektrális szintézisre, azzal a különbséggel, hogy itt csak a húrt modellezzük lecsengő szinuszos hullámok összegeként, a gerjesztés (pengetés) és a hangszertest által definiált szűrő továbbra is megmarad. Előnye, hogy egyszerűen implementálható módusonként egy vagy két lecsengő rezonátorrendszerrel, viszont cserébe számításigényesebb.

Rugalmassága és testreszabhatósága miatt a szakdolgozatomhoz ezt a húrmodellt választottam, így a következő fejezetben fejtem ki részletesen.

3.3 Modális húrmodell

A véges differenciák módszere közvetlenül a differenciálegyenletet diszkretizálja, a digitális hullámvezető pedig a haladóhullám-megoldást időtartományban.

Ezzel ellentétben a modális húrmodell a frekvenciatartományból közelíti meg a problémát. A modellezendő rendszert másodfokú lecsengő rendszerekből állítja össze, ezzel minden módust külön lehet paraméterezni, ami a legnagyobb rugalmasságot, és elvben a legnagyobb pontosságot eredményezi. Megfelelő minőségű mérések és analízis segítségével a valódi húr igen jó közelítését kaphatjuk, ezért esett erre a módszerre a választás. Korábban problémát okozott a modális húrmodell nagyobb számításigénye, de a mai számítógépekben található mikroprocesszorok számítási teljesítménye elegendő több száz, több ezer módus implementálásához is.

3.3.1 Folytonos időbeli egyenletek

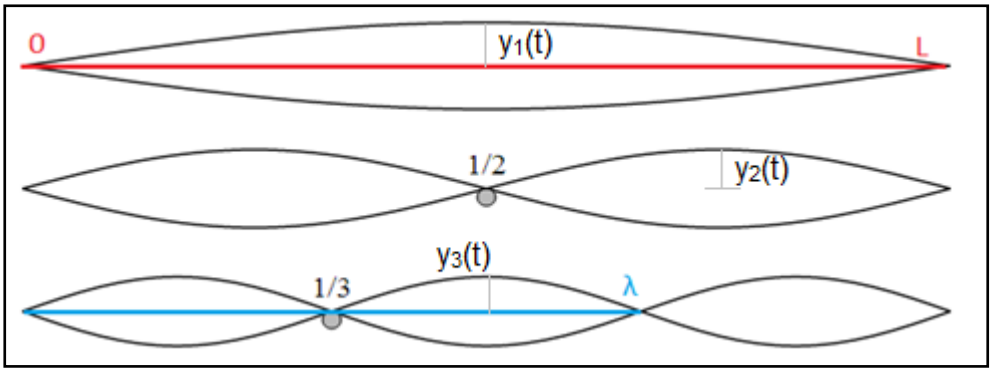
A levezetéseket [Bank 2006] és [Bank 2010] tartalmazza. Induljunk ki megint a (3.2.1) differenciálegyenletből:

$$\mu \frac{\partial^2 y}{\partial t^2} = T_0 \frac{\partial^2 y}{\partial x^2} - ES\kappa^2 \frac{\partial^4 y}{\partial x^4} - 2R\mu \frac{\partial y}{\partial t} + d_y(x,t), \qquad (3.2.1)$$

valamint tekintsük a két végén ($x = 0$ és $x = L$) rögzített húr alakjára vonatkozó egyenletet:

$$y(x,t) = \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right), x \in [0, L] \qquad (3.3.1)$$

ahol $y_k(t)$ a k-adik módus pillanatnyi amplitúdója. 3.3-1. ábra szemlélteti az egyenlet jelentését. Jól látható, hogy ha ismerjük $y_k(t)$ értékét a kívánt k módusokra az adott időpillanatban, a húr alakját elég jó pontossággal ismerjük. Mivel végtelen sok módust nem tudunk kezelni, meg kell szabni azt a maximális módusszámot, ahol még van értelme kiszámolni az adott módushoz tartozó hullámformát, és hallhatóan nem romlik a minőség.



3.3-1. ábra: két végén rögzített húron létrejövő hullámformák [Wikipedia 1]

$y_k(t)$ módusamplitúdó-sorozat előállítása érdekében behelyettesítjük (3.3.1)-et (3.2.1)-be:

$$\mu \frac{\partial^2 \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right)}{\partial t^2}$$

$$= T_0 \frac{\partial^2 \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right)}{\partial x^2} - ES\kappa^2 \frac{\partial^4 \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right)}{\partial x^4}$$

$$- 2R\mu \frac{\partial \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right)}{\partial t} + d_y(x,t).$$

Szorozzuk meg az egyenletet $\sin\left(\frac{n\pi x}{L}\right)$ taggal, majd integráljuk *x* szerint a [0:L] tartományon:

$$\mu \frac{\partial^2}{\partial t^2} \int_0^L \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right) \sin\left(\frac{n\pi x}{L}\right) dx$$

$$= T_0 \frac{\partial^2}{\partial x^2} \int_0^L \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right) \sin\left(\frac{n\pi x}{L}\right) dx$$

$$- ES\kappa^2 \frac{\partial^4}{\partial x^4} \int_0^L \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right) \sin\left(\frac{n\pi x}{L}\right) dx$$

$$- 2R\mu \frac{\partial}{\partial t} \int_0^L \sum_{k=1}^{\infty} y_k(t) \sin\left(\frac{k\pi x}{L}\right) \sin\left(\frac{n\pi x}{L}\right) dx$$

$$+ \int_0^L d_y(x,t) \sin\left(\frac{n\pi x}{L}\right) dx,$$

Zygmund [2002] szerint a trigonometrikus függvényrendszer ortogonális:

$$\int_0^L \sin\left(\frac{k\pi x}{L}\right)\sin\left(\frac{n\pi x}{L}\right)dx = \begin{cases} 0, k \neq n \\ \frac{L}{2}, k = n \end{cases}$$

Ezt figyelembe véve:

$$\mu\frac{\partial^2 y_n(t)}{\partial t^2}\frac{L}{2} = T_0\frac{\partial^2 y_n(t)}{\partial x^2}\frac{L}{2} - ES\kappa^2\frac{\partial^4 y_n(t)}{\partial x^4}\frac{L}{2} - 2R\mu\frac{\partial y_n(t)}{\partial t}\frac{L}{2} + \int_0^L d_y(x,t)\ \sin\left(\frac{n\pi x}{L}\right)dx,$$

ezt átrendezve, a változókat összevonva, és $y_n(t) \rightarrow y_k$ helyettesítéssel élve:

$$\frac{\partial^2 y_k}{\partial t^2} + a_{1,k}\frac{dy_k}{dt} + a_{0,k}\,y_k = b_{0,k}F_{y,k}(t), \qquad (3.3.2)$$

ahol:

- $a_{1,k} = 2R_k$ (3.3.2a)
- $a_{0,k} = \frac{T_0}{\mu}\left(\frac{k\pi}{L}\right)^2 + \frac{ES\kappa^2}{\mu}\left(\frac{k\pi}{L}\right)^4$ (3.3.2b)
- $b_{0,k} = \frac{2}{L\mu}$ (3.3.2c)
- $F_{y,k}(t) = \int_{x=0}^{L}\sin\left(\frac{k\pi x}{L}\right)d_y(x,t)dx.$ (3.3.2d)

Mivel most már módusonként tudjuk a húrmodellt vizsgálni, volt értelme bevezetni R_k-t a frekvenciafüggő veszteségek modellezésére.

A későbbiekben olyan gerjesztőerőt fogunk használni, amely csak a húr x_0 pontjára hat. Erre felírva (3.3.2d) egyenletet:

$$F_{y,k}(t) = \int_{x=0}^{L}\delta(x - x_0)\sin\left(\frac{k\pi x}{L}\right)d_{y,x_0}(t)dx = \sin\left(\frac{k\pi x_0}{L}\right)d_{y,x_0}(t). \qquad (3.3.2e)$$

A (3.3.2) differenciálegyenlet megoldására $F_{y,k}(t) = \delta(t)$ impulzusgerjesztés mellett, zérus kezdeti értékekkel egy exponenciálisan lecsengő szinusz adódik, [Bank 2010] mintájára:

$$y_{\delta,k}(t) = A_k e^{-\frac{t}{\tau_k}}\sin(2\pi f_k t), \qquad (3.3.3)$$

ahol:

- $A_k = \frac{b_{0,k}}{2\pi f_k} = \frac{1}{\pi L\mu f_k}$ (3.3.3a)

- $\tau_k = \frac{2}{a_{1,k}} = \frac{1}{R_k}$ (3.3.3b)

- $f_k = \frac{1}{2\pi}\sqrt{a_{0,k} - \frac{a_{1,k}^2}{4}} \approx f_0 k\sqrt{1 + Bk^2}$, (3.3.3c)

ahol A_k a kezdeti amplitúdó, τ_k a lecsengési idő, f_k a *k*-adik módusfrekvencia, f_0 a húr alapfrekvenciája, B az inharmonicitási együttható. A valós implementációban f_k módusfrekvenciákat közvetlenül a modellezendő hangszer hangjának spektrumából határozzuk meg, csakúgy, mint a τ_k lecsengési időket.

Mivel $F_{y,k}(t) = \delta(t)$ impulzusgerjesztésre határoztuk meg a választ, tetszőleges gerjesztőerőre időtartománybeli konvolúcióval határozhatjuk meg:

$$y_k(t) = y_{\delta,k}(t) * F_{y,k}(t) \qquad (3.3.4)$$

Összesítve (3.3.2d), (3.3.3), (3.3.4) és (3.3.1) egyenleteket, tetszőleges $d_y(x,t)$ gerjesztőerő-eloszlásra a húrformát bármely időpillanatra a következőképpen kaphatjuk:

- $d_y(x,t)$-ból $F_{y,k}(t)$ adott módusra ható erőt számolunk (3.3.2d) alapján,
- (3.3.3)-ban kapott impulzusválaszt konvolváljuk a módusokra ható erővel (3.3.4) szerint, ebből megkapjuk a vizsgált módusok pillanatnyi amplitúdóját,
- (3.3.1) szerint a módusalakok és a pillanatnyi amplitúdójuk összegzéséből megkapjuk a húralakot.

Gitár esetében a húrformán kívül fontos a húrlábra ható erő is, ugyanis ez továbbítja a rezgést a hangszertest felé. Bank [2010] szerint az $x = 0$ pontban ható $F_b(t)$ erő:

$$F_b(t) = T_0 \left.\frac{\partial dy}{dx}\right|_{x=0} = \frac{T_0\pi}{L}\sum_{k=1}^{\infty} k y_k(t), \qquad (3.3.5)$$

ami $y_k(t)$ pillanatnyi módusamplitúdók súlyozott összege.

3.3.2 Diszkretizáció

A kapott eredményeket diszkretizálni kell a szoftveres implementációhoz. Mivel minden egyenletet időtartományban vezettünk le, ez nem okoz problémát. Azonban többféle transzformáció is létezik, például impulzus invariáns transzformáció, bilineáris transzformáció [Rabiner és Gold 1975].

Bank [2006] az impulzus invariáns transzformációt választja, mert ebben az esetben a diszkrétidejű impulzusválaszban megjelenik egy egyszeres késleltetés. Ezzel kiküszöböljük a késleltetés nélküli visszacsatolást, ami bonyodalmakat okozhat az implementációban.

Az impulzustartó transzformáció alapja, hogy a diszkrétidejű rendszert úgy állítjuk elő a folytonos idejű rendszerből, hogy a DI rendszer impulzusválasza a FI rendszer impulzusválaszából mintavételezéssel kapott jellel egyezzen meg. Tehát a keresett DI rendszer impulzusválasza (3.3.3) mintavételezésével:

$$y_{\delta,k}[n] = \frac{1}{f_s} y_{\delta,k}(t_n) = \frac{1}{f_s} A_k e^{-\frac{t_n}{\tau_k}} \sin(2\pi f_k t_n), \qquad (3.3.6a)$$

$$y_{\delta,k}[n] = \frac{1}{f_s} A_k e^{-\frac{n}{\tau_k f_s}} \sin\left(2\pi \frac{f_k}{f_s} n\right), \quad (3.3.6b)$$

ahol $t_n = nT_s$, $T_s = 1/f_s$ mintavételi idő.

Az impulzusválaszban megjelenik egy $\frac{1}{f_s}$ szorzás. Ennek magyarázata az, hogy míg a folytonos idejű Dirac-impulzus területe 1, addig diszkrét időben ez $\frac{1}{f_s}$. Emiatt, hogyha a DI rendszert egységimpulzussal gerjesztjük, annak érdekében, hogy ténylegesen impulzus invariáns legyen a transzformáció, be kell iktatni az $\frac{1}{f_s}$ szorzást a diszkrét idejű impulzusválaszba. A diszkrét rendszer paramétereinek meghatározásához Z-transzformációt használunk (3.3.6b) DI impulzusválaszon. Jury [1964]-ből és a Z-transzformáció tulajdonságaiból tudjuk, hogy

$$Z\{\alpha^n \sin(\omega_0 n)\, \varepsilon[n]\} = \frac{\alpha z^{-1} \sin(\omega_0)}{1 - 2\alpha z^{-1} cos(\omega_0) + \alpha^2 z^{-2}}, \qquad (3.3.7)$$

ahol $\varepsilon[n]$ az egységugrást jelenti. Korábban is feltételeztük, hogy csak $t > 0$ tartományra értelmezzük az egyenleteket, ezért elhagytuk az $\varepsilon[n]$, $\varepsilon(t)$ szorzótagokat, és a továbbiakban sem írjuk ki, de a Z-transzformáció tulajdonságaihoz értelmezéséhez szükséges. A (3.3.7) egyenlet alapján (3.3.6b) Z-transzformáltja:

$$Z\left\{\frac{1}{f_s} A_k e^{-\frac{n}{\tau_k f s}} \sin\left(2\pi \frac{f_k}{fs} n\right)\right\} = \frac{\frac{A_k}{f_s} e^{-\frac{1}{\tau_k f s}} z^{-1} \sin\left(2\pi \frac{f_k}{fs}\right)}{1 - 2e^{-\frac{1}{\tau_k f s}} z^{-1} \cos\left(2\pi \frac{f_k}{fs}\right) + e^{-\frac{2}{\tau_k f s}} z^{-2}}$$

Az együtthatók egy komplex szám képzetes és valós részeiből adódnak, így egyszerűsítések és átírás után [Bank 2010] mintájára:

$$H_{res,k}(z) = \frac{b_k z^{-1}}{1+a_{1,k}z^{-1}+a_{2,k}z^{-2}} \qquad (3.3.8)$$

ahol:

- $b_k = \frac{A_k}{fs} Im\{p_k\}$
- $a_{1,k} = -2Re\{p_k\}$
- $a_{2,k} = |p_k|^2$
- $p_k = e^{j2\pi\frac{f_k}{f_s}} e^{-\frac{1}{\tau_k f_s}}$.

Ez a rendszer már közvetlenül implementálható késleltetésekkel és erősítésekkel. Mivel az impulzusválaszt csak egyetlen módusra néztük, minden kívánt módusra meg kell valósítani egy ilyen másodfokú rendszert, és a válaszaik összege adja a kimenő jelet. Az implementációhoz szükséges egyenlethez inverz z-transzformáljuk és átrendezzük (3.3.8) átviteli függvényt:

$$y_{\delta,k}[n] = b_k F_{y,k}[n-1] - a_{1,k} y_{\delta,k}[n-1] - a_{2,k} y_{\delta,k}[n-2]. \qquad (3.3.9)$$

A (3.3.3a) egyenletet is figyelembe véve látszik az is, hogy a húrmodellnek valójában csak kettő mérendő, módusra jellemző paramétere van: f_k módusfrekvencia és τ_k lecsengési idő. Az A_k módusamplitúdókat (3.3.3a) alapján számoljuk. Tehát a módusfrekvenciák és lecsengési idők ismeretében már tudjuk modellezni a húrt. Minél több módust implementálunk, annál pontosabban lesz a modell, viszont a számítási igény is nő.

Az is megfigyelhető, hogy $F_{y,k}[n]$ nem szerepel az egyenletben, ami annak eredménye, hogy impulzus invariáns transzformációt választottunk. Ez azért szerencsés, mert később a húr-pengető kölcsönhatás számításánál a húrra csak a gerjesztőerő előző értéke hat, így azt a cikluson belül bárhol számíthatjuk.

3.3.3 A második transzverzális polarizáció figyelembe vétele

Az eddigi egyenletek felírásánál feltételeztük, hogy a húr csak egy síkban rezeg. Ez a valóságban nincs így, a húr rezgése két transzverzális polarizációjú rezgés összegeként írható fel [Bank 2006]. Ezt két külön rezonátorbankként implementálhatjuk: a húrmodell kimenetében két rezgő rendszer, két párhuzamosan kapcsolt csatolt húrmodell kimeneteinek összegét kell számolni. A másodlagos

rezonátorok módusfrekvenciái közel esnek az főrezonátorok módusfrekvenciáihoz, és ezzel egy lebegés jellegű hanghatást állítanak elő. Ez azonban a szintetizátor működéséhez nem elengedhetetlenül szükséges, ezért a másodlagos rezonátorokra a későbbiekben is csak mint kiegészítő tagokra tekintünk, amelyek adott esetben el is hagyhatók.

Emiatt a húrmodell gerjesztéssel való kölcsönhatásában sem vesznek részt: noha a másodlagos rezonátorok ugyanazt a gerjesztőerőt kapják meg, mint a főrezonátorok, visszahatás a gerjesztésre csak a főrezonátorok felől történik.

A konkrét implementációról a 4.3 fejezetben lesz szó.

3.4 Gerjesztésmodell

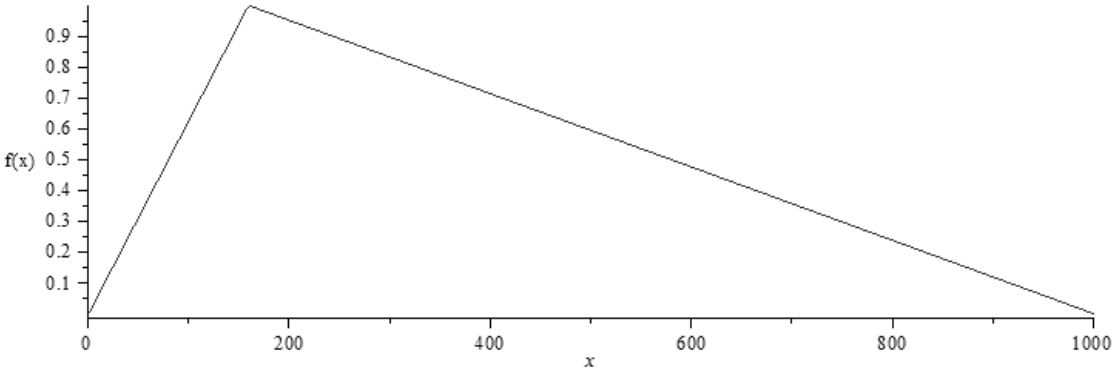
A gitármodellben a húrmodell gerjesztésére két módszer is adódik. Egyrészt gerjeszthetjük (3.3.9) alapján, amelyhez meg kell határozni az $F_{y,k}[n]$ gerjesztőerőt. Ezt a módszert használom a pengetőmodellezésnél, ahol a pengetőt magát is modellezem, és a húrmodell és a pengetőmodell közötti kölcsönhatás adja a gerjesztést.

Gerjeszthetjük a húrmodellt úgy is, hogy csak a kezdeti értékeit állítjuk be, és abból a helyzetből elindítjuk, nulla $F_{y,k}[n]$ mellett. Az ujjal történő pengetés modellezésére ezt használom, mert ilyenkor a gitár csak azután szólal meg, hogy az ujj elhagyta a húrt. Tehát jogosan feltételezhetjük, hogy csak az elengedéskor fennálló húrforma számít, ami előtte történik, nem.

3.4.1 Ujjal pengetés

Ujjal pengetésnél először beállítjuk a húr alakját, majd magára hagyjuk. Az ujjat magát a húr egyetlen pontjára hatónak tekintjük, így a létrejövő húralak egy háromszög, amelynek csúcsai *0, L* és x_0 pontban vannak, ahol x_0 a gerjesztés helye (lásd 3.4-1. ábra). A háromszög magassága legyen pontosan *1*. Írjuk fel ezt a függvényt a [0:L] tartományon ablakozott lineáris jelek összegeként:

$$f(x) = \varepsilon(x)\left(\frac{x}{x_0}\right) + \varepsilon(x-H)\left(\frac{-x+L}{L-x_0} - \frac{x}{x_0}\right) - \varepsilon(x-L)\left(\frac{-x+L}{L-x_0}\right) \qquad (3.4.1)$$



3.4-1. ábra: húrforma x_0 = 0.16L pontban, A = 1 magasságra gerjesztve

A (3.4.1) függvényt kell megkapnunk a húrformára vonatkozó (3.3.1) egyenlettel. Az így adódó $y_k = y_k[0]$ sorozat adja meg a kívánt kezdeti értékeket. A kettőt egyenlővé téve:

$$\sum_{k=1}^{\infty} y_k \sin\left(\frac{k\pi x}{L}\right) = \varepsilon(x)\left(\frac{x}{x_0}\right) + \varepsilon(x-x_0)\left(\frac{-x+L}{L-x_0} - \frac{x}{x_0}\right) - \varepsilon(x-L)\left(\frac{-x+L}{L-x_0}\right)$$

Szorozzuk meg mindkét oldalt $\sin\left(\frac{n\pi x}{L}\right)$ taggal és integráljunk x szerint 0-tól L-ig:

$$\int_0^L \sum_{k=1}^{\infty} y_k \sin\left(\frac{k\pi x}{L}\right)\sin\left(\frac{n\pi x}{L}\right)dx = \int_0^L \frac{x}{x_0}\sin\left(\frac{n\pi x}{L}\right)dx$$
$$+ \int_{x_0}^L \left(\frac{-x+L}{L-x_0} - \frac{x}{x_0}\right)\sin\left(\frac{n\pi x}{L}\right)dx - \int_L^L \left(\frac{-x+L}{L-x_0}\right)\sin\left(\frac{n\pi x}{L}\right)dx$$

Használjuk ki (3.3.2) megoldásánál is alkalmazott trigonometrikus függvényrendszer ortogonalitását a bal oldalon, végezzük el az integrálást a jobbon, valamint vezessük be a $D = \frac{x_0}{x}$ relatív pengetési pozíció jelölést, és az egyszerűsítések és összevonások elvégése után:

$$y_k[0] = \frac{4L(\mathrm{D}\sin(\pi k) - \sin(\pi k D))}{\pi D k\,(D-1)\,2\pi k L} \qquad (3.4.2)$$

Mivel másodfokú a rendszer, szükség van a nulladik elem előtt eggyel fennálló $y_k[-1]$ értékekre is. Az egyszerűség kedvéért feltételezzük, hogy a negatív időtartományban csillapítatlanul rezgett a húr, valamint hogy a nulladik időpillanatban lép fel a legnagyobb kitérés. Az első feltétel miatt az impulzusválaszból elhagyjuk az

exponenciális tagot, a második pedig azt jelenti, hogy *cos* függvény szerepel az egyenletben:

$$y_k[-1] = y_k[0]cos(-2\pi f_k\, dt) = y_k[0]cos\left(-2\pi\frac{f_k}{f_s}\right). \qquad (3.4.3)$$

A háromszög alakú húr ilyen módú leírása egy Fourier-sorhoz hasonlatos sorba fejtés, és az is nyilvánvalóan igaz rá, hogy a magasabb harmonikusok felelősek az éles átmenetekért, így kevés harmonikus használata esetén a háromszög sarka ’letompul’. Ez a kimenetben a magasfrekvenciás tagok hiányát okozza.

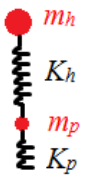
3.4.2 Pengetővel pengetés

Pengetővel történő pengetésnél a pengetőt is modellezni kell, és szimulálni kell a húrmodellel történő kölcsönhatását. Az így adódó erőjel szolgál a húrmodell bemeneteként.

A pengetőmodellre a következő feltételeknek kell teljesülniük:

- a pengető és a húr anyaga között kismértékű rugalmas kölcsönhatás léphet fel
- a pengető anyaga hajlékony, és a húrra kifejtett erő a lehajlítás mértékének függvénye
- bizonyos erő felett a pengető lecsúszik a húrról, magára hagyva azt
- a pengető is szabadon rezeg a húr elhagyása után (de ennek már nincs hatása a kimeneti jelre)
- figyelembe kell venni a pengető kéz és alkar tömegét is
- feltesszük, hogy a gerjesztőerő csak egy pontban hat a húrra.

Ennek megfelelően a választott modell két tömegpontból és két rugóból áll (3.4-2. ábra).



3.4-2. ábra: pengetőmodell

Az m_h tömegpont jelképezi a zenész kezét, m_p a pengetőt magát. K_h rugóállandójú rugó a pengető lehajlásából fellépő erőt modellezi, K_p pedig a pengető felületi rugalmasságát.

Ennél pontosabb modell lenne Perng et al. [2010] által bemutatott módszer, amely egy csembalópengetőt modellez. Annak érdekében, hogy a húrt két síkban pendítse, pontosan felírja a kétdimenziós pengető lehajlását, és ebből számol tovább. Ebben a dolgozatban az implementációnál más módon valósítottuk meg a húr két síkban rezgését (lásd 4.3. fejezet), amihez Perng modellje nehezebben illeszthető, ezért a lényegesen egyszerűbb rugós modellre esett a választás.

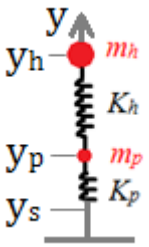
Az egyszerűség kedvéért a rugóerőket is az összenyomás lineáris függvényeként adjuk meg. Ezt megtehetjük, mert pengetővel való játéknál kevésbé érdekes, hogy mi történik, mielőtt elhagyja a pengető a húrt – részint, mert a létrejövő hang elhanyagolható hangerejű a tényleges megpengetett hanghoz képest, részint pedig mert nagyon rövid ideig tart.

Összességében tehát ez is egyfajta kezdetiérték-beállítás, hiszen a pengető lecsúszása utáni hang a lényeges, de a valósághűség és a fizikai jelleg megtartása érdekében szimuláljuk az ez előtti eseményeket is.

Az implementációhoz induljunk ki az l hosszra összenyomott, kezdetben l_0 hosszú, K rugóállandójú rugó által kifejtett erő egyenletéből:

$$F = -K(l - l_0) \qquad (3.4.4)$$

A 3.4-2. ábrán vázolt elrendezésre vegyünk fel egy koordinátarendszert a 3.4-3. ábrán látható módon:



3.4-3. ábra: pengetőmodell koordinátarendszerrel

Ahol y_h a kéz (azaz a pengető „szélesebbik” fele) helye, y_p a pengető húr felőli („hegyes”) felének pozíciója, y_s a húr kitérése az adott pontban. A (3.4.4) összefüggést felírva a két rugóra, valamint $l_0 = 0$ választással élve:

$$F_h = -K_h(y_h - y_p) \qquad (3.4.5a)$$

$$F_p = -K_p(y_p - y_s) \qquad (3.4.5b)$$

$$F_{s,x_0} = -F_p, \qquad (3.4.5c)$$

ahol F_h a pengető lehajlásából eredő erő, F_p a pengető felületi rugalmasságából eredő, pengető és húr között ható erő, F_{s,x_0} a húr x_0 pontjára ható erő. Az $l_0 = 0$ választás jogosságát indokolja, hogy a pengetés pillanatában a pengető közel merőlegesen áll a húrra, tehát a húr függőleges síkjára vett vetületén $y_h(0) = y_p(0) = y_s(0)$.

A további számításokhoz feltételezzük, hogy a pengető a *t=0* időpillanatban lép kölcsönhatásba a húrral, előtte állandó v_0 sebességgel halad a koordinátarendszeren negatív irányba, és egyik rugó sincs terhelve, tehát $v_p(0) = v_h(0) = v_0$. A húr kezdetben $y_s(0)=0$ kitéréssel rendelkezik.

Vegyük a következő newtoni egyenleteket:

$$\Delta v = \frac{1}{m}\int_{t_1}^{t_2} F(t)dt \qquad (3.4.6a)$$

$$\Delta y = \int_{t_1}^{t_2} v(t)dt \qquad (3.4.6b)$$

Ebből az alakból egy lépésben diszkretizálhatunk, és tegyük ezt meg (3.4.5a) összefüggésekkel is:

$$F_h[n] = -K_h(y_h[n-1] - y_p[n-1]) \qquad (3.4.7a)$$

$$v_h[n] = v_h[n-1] + \frac{1}{m_h f_s} F_h[n] \qquad (3.4.7b)$$

$$y_h[n] = y_h[n-1] + \frac{1}{f_s} v_h[n] \qquad (3.4.7c)$$

Látszik, hogy v_h számításánál F_h aktuális értéke is szükséges, és hasonlóképpen y_h számításánál v_h aktuális értéke, ezért a kiszámításuk sorrendje sorrend kötött.

Felírhatjuk ugyanezt (3.4.5b)-re is:

$$F_p[n] = -K_p(y_p[n-1] - y_s[n-1]) \qquad (3.4.8a)$$

$$v_p[n] = v_p[n-1] + \frac{1}{m_p f_s} F_p[n] \qquad (3.4.8b)$$

$$y_p[n] = y_p[n-1] + \frac{1}{f_s} v_p[n] \qquad (3.4.8c)$$

(3.4.7) és (3.4.8) összefüggésekkel le tudjuk írni a pengető viselkedését attól a pillanattól, hogy megérinti a húrt, addig a pillanatig, amíg elhagyja azt. Arról, hogy a pengető mikor csúszik le a húrról, ugyanebben a fejezetben később lesz szó.

Az implementációhoz meg kell még adni a kezdetiértékeket, amelyekről már esett szó:

$$y_h[0] = y_p[0] = y_s[0] = 0 \qquad (3.4.9a)$$

$$v_h[0] = v_p[0] = v_0 \qquad (3.4.9b)$$

$$F_h[0] = F_p[0] = F_s[0] = 0 \qquad (3.4.9c)$$

Ezután már csak a paraméterek meghatározása van hátra, és a modell implementálható MATLAB-ban prototípusként, és C++ nyelven a VST plugin részeként is.

A pengető és a kéz tömegét (m_p*=0.005 kg,* m_h*=0.35 kg*) paraméterbecsléssel határoztam meg, csakúgy, mint a rugóállandókat (K_p*=80 N/m,* K_h*=400 N/m*). A kezdeti v_0 sebesség átlagos értékének meghatározásához a következő módszert alkalmaztam: közepes pengetési sebességgel mozgattam a kezem fel-le a gitár előtt, a pengetett húr fölött kb. 1 cm-rel indítva és az alatt kb. 1 cm-rel megállítva a kezemet. Megmértem a pengetés átlagos frekvenciáját, ami kb. 2 Hz-re jött ki. Ebből kijött, hogy amikor a pengető a húrhoz ér, átlagosan *0.125 m/s* sebességgel halad.

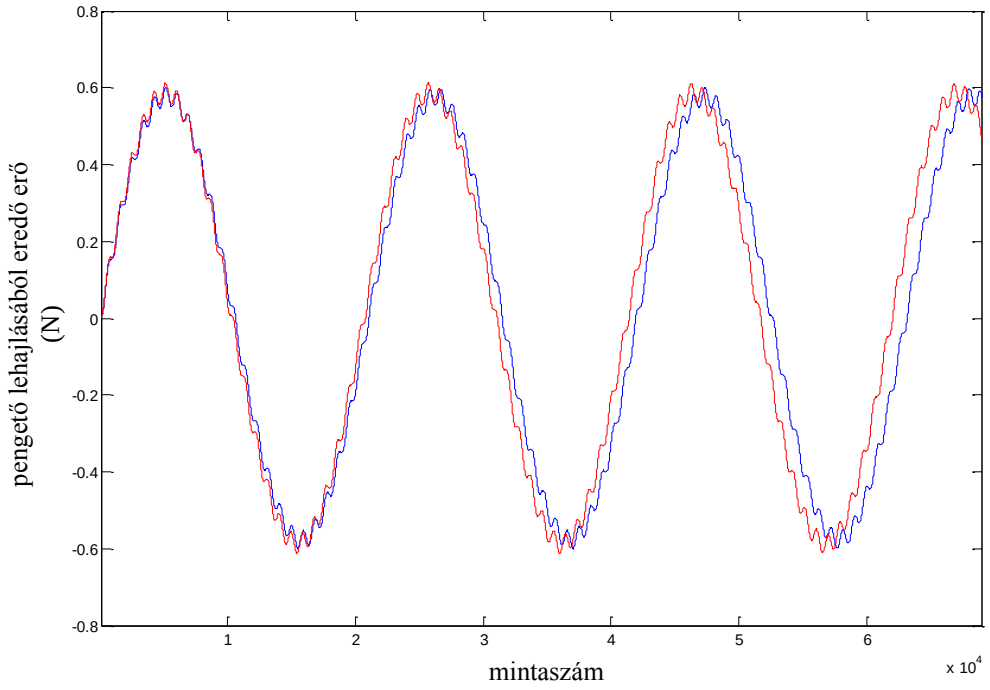
A pengető paramétereinek meghatározásánál egy kevéssé fizikai, inkább intuitív módszert alkalmaztam. Mivel azonban a pengető-húr kölcsönhatás csak onnantól lényeges, hogy a pengető elhagyta a húrt, a paramétereket elegendő becsülni, és a kapott eredményhez később hozzáhangolni azt az erőt, ahol a pengető lecsúszik a húrról.

Ez a határerő az a kritikus erőérték, aminél a pengető és a húr között fellépő tapadási súrlódás nem elegendő ahhoz, hogy együtt maradjanak, és a pengető elhagyja a húrt, ami onnantól kezdve magára hagyva rezeg. Minél nagyobb ez a határerő, annál jobban megfeszül a húr, mielőtt megpendül, tehát annál hangosabb lesz a hang. A valóságban ez úgy jelenik meg, hogy halk gitárjátéknál a zenész a pengető megdöntésével elősegíti annak lecsúszását, míg erőteljes játéknál közel merőlegesen

tartja a húrra, és a pengető erősebb szorításával biztosítja, hogy a húr kellőképpen megfeszüljön.

Ez a paraméter tehát ideális ahhoz, hogy a dinamikát modellezzük vele. A megállapított paraméterek beállítása után a szimulációból (3.4-4. ábra) látszik, hogy a legnagyobb lehajlási erő (ezekkel a paraméterekkel) ~0.61 N, ezután a pengető „visszafordul", és mintha a húrral egy rendszert alkotnának (a pengető és a húr „összetapadnak"), periodikusan rezeg. Így amikor majd a MIDI adatvektor *velocity* tagjával állítjuk a pengetés dinamikáját, akkor ésszerű döntésnek tűnik, hogy a legnagyobb *velocity* értékhez tartozzon a legnagyobb határerő, amelynek értéke 0.6 N.

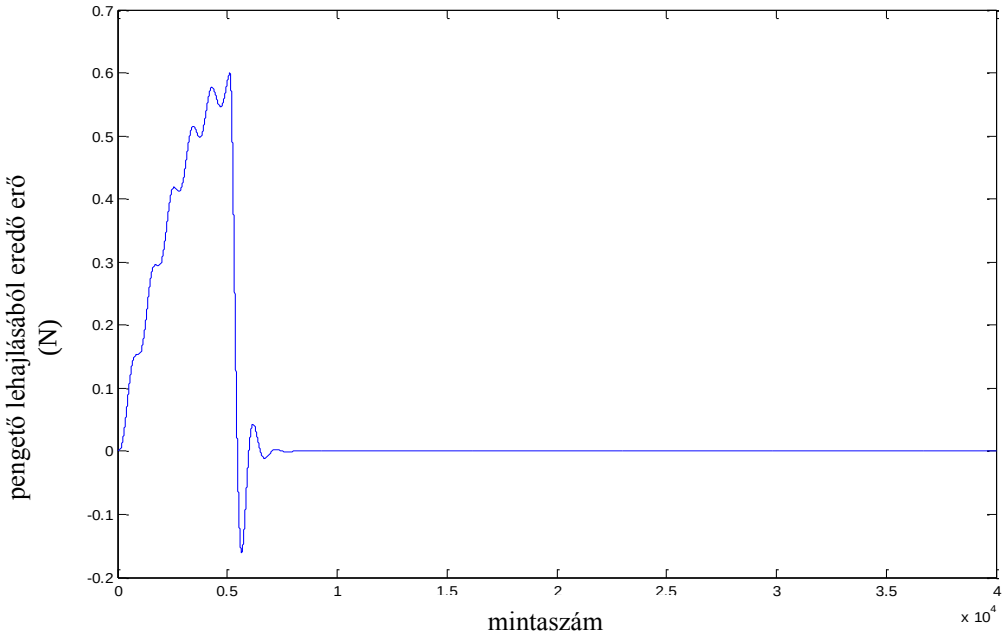
Fontos megjegyezni, hogy ez az érték nem egy fizikai vagy matematikai levezetés eredménye, hanem a használt paraméterekkel szimulált modellt elemezve állapítottam meg.



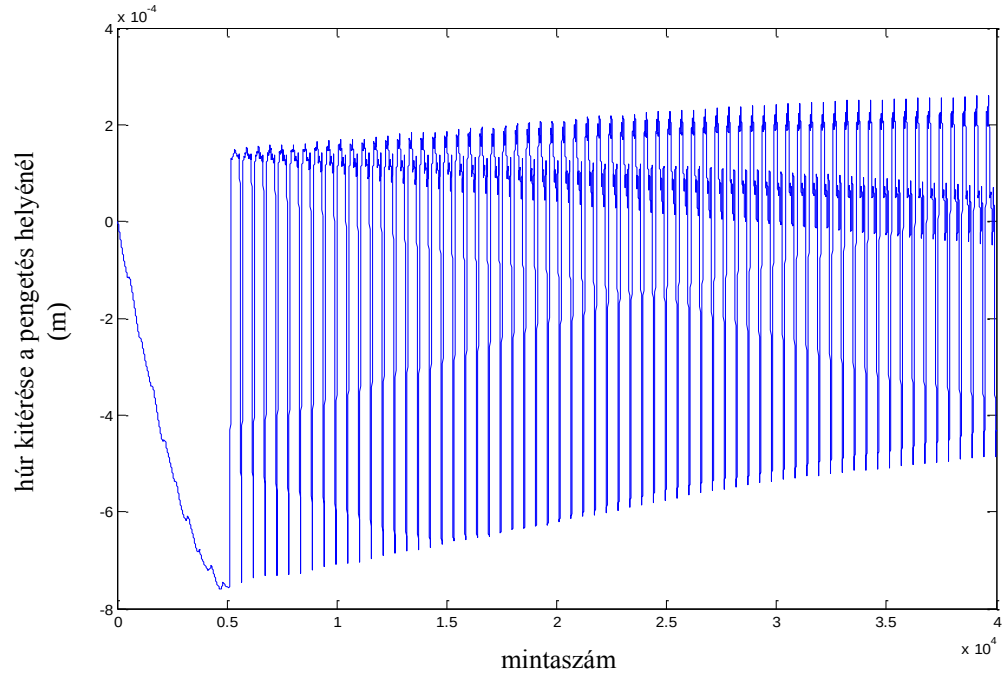
3.4-4. ábra: pengető lehajlásából eredő erő (F_h), ha a pengető sosem hagyja el a húrt; kék: F_h erő az üres mély E húrra, piros: F_h erő az üres A húrra

A húr elhagyása után a pengető (és a húr is) szabadon rezeg (3.4-5. és 3.4-6. ábrák). A pengető lecsengő rezgéséért egy csillapítást jelentő erő felel, amely a légellenállást és a pengető anyagában fellépő melegedést modellezi. Mivel a húr megpendülése után a pengető viselkedésével nem foglalkozunk, ennek a csillapításnak

nincs jelentősége, és a valós idejű implementációban nem is szerepel, de a MATLAB-ban írt prototípusba ellenőrzés céljából belekerült. A mély *E* húr megpengetett x_0 pontjának kitérése látható az idő (mintaszám) függvényében a 3.4-6. ábrán.



3.4-5. ábra: pengető lehajlásából eredő erő (F_b), ha 0.6 N határerőt elérve a pengető elhagyja a húrt (mély E húr üresen pengetve)



3.4-6. ábra: húr kitérése (Y_S) a pengetés helyénél (mély E húr üresen pengetve). A megpendítés kb. az ötezredik mintánál történik.

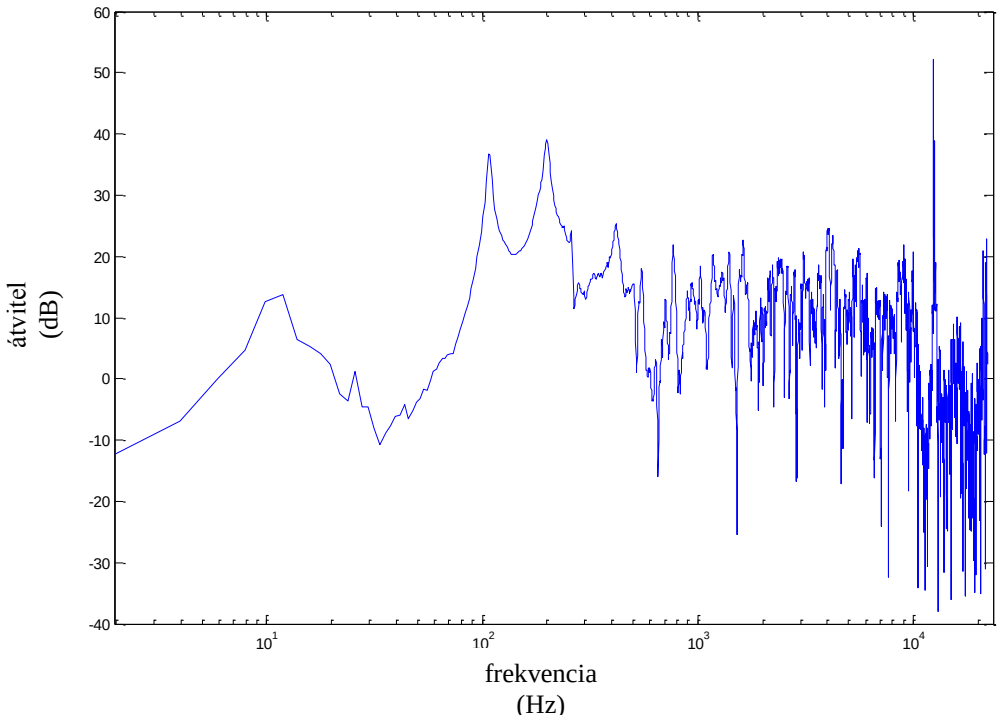
3.5 Gitártestmodell

Akusztikus gitár esetén a test a hang kisugárzásának központi eleme. A gitár hangzását nagyban befolyásolja a test formája, mérete, anyaga, a hanglyuk pozíciója, mérete, a nyak rögzítési módja, valamint a test belső felépítése. A húrok és a test közötti erőátvitelt a húrláb (bridge) biztosítja, amelynek formája, kialakítása, anyaga szintén befolyásolja a hangszínt.

Nyilvánvaló, hogy a húrláb és a gitártest együttese a húr rezgéseinek módusfrekvenciáit, amplitúdóit, valamint a lecsengéseket is befolyásolja. A modellben ezek közül azonban csak a test frekvenciafüggő erősítését vesszük közvetlenül figyelembe.

A módusfrekvenciákat és a lecsengési időket a húrmodellben állítjuk be, méréssel meghatározott paraméterek alapján. A következő fejezetben lesz róla szó, hogy a mérésekhez nem szereltük le a húrokat a gitárról, így azokat megpendítve mind a frekvenciák, mind a lecsengési idők, továbbá az amplitúdók is a gitártest által már módosított értékek. Ezzel a gitártest hatása a módusfrekvenciákra és a lecsengési időkre a modellben szereplő hangszertest blokk helyett a húrmodellbe kerül. Azonban az amplitúdókat nem a mérésből, hanem (3.3.3a) egyenlet alapján állítjuk be, tehát ezeket még meg kell szűrni a gitártesttel, hogy a kimeneti amplitúdók a valóságnak megfelelőek legyenek.

Ezek szerint a gitártestet modellezhetjük egy nagy fokszámú szűrővel, amely a húrmodell kimeneti jelét szűri meg. A 3.5-1. ábrán látható a gitártest mérésből számított átviteli függvénye, a mérés körülményeiről a 4. fejezetben lesz szó.



3.5-1. ábra: gitártest mért átviteli függvénye

A hagyományos FIR és IIR szűrők azonban lineáris frekvenciafelbontásúak, szemben az emberi hallás logaritmikus jellegű felbontásával. A hangszertestet reprezentáló szűrő logaritmikus felbontásának megvalósítására Bank [ICMC07] több megoldást is felsorol.

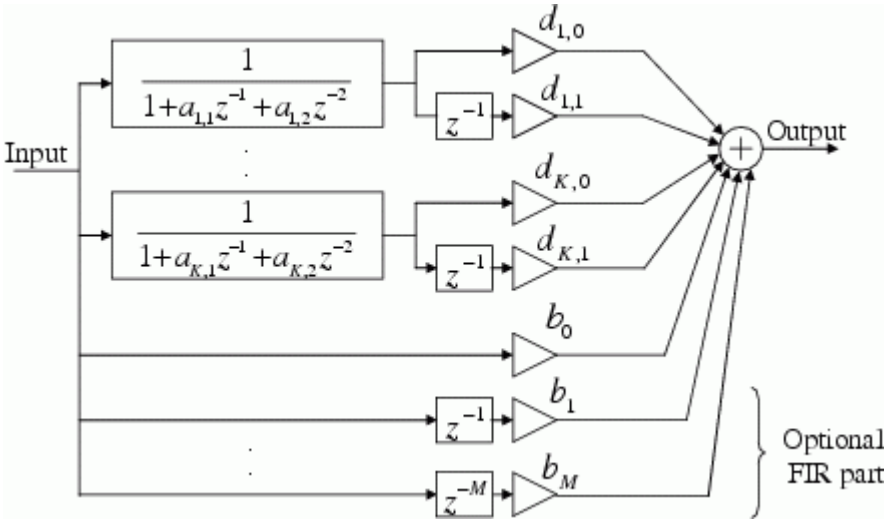
A warpolt szűrő kiválóan alkalmas hangszertest megvalósítására, mert igen jól tudja közelíteni az emberi hallás frekvenciafelbontását [Karjalainen és Smith 1996]. A módszer lényege, hogy a hagyományos FIR vagy IIR szűrőkben szereplő z^{-1} késleltetéseket azonos mindentáteresztő szűrőkkel helyettesíti:

$$z^{-1} \leftarrow D(z) = \frac{z^{-1}-\lambda}{1-\lambda z^{-1}}. \qquad (3.5.1)$$

A warpolt szűrő frekvenciafelbontása a λ paraméterrel hangolható. Ezzel a megoldással lényegesen kisebb fokszámú szűrőt elég illeszteni, például klasszikus gitár esetén 500-2000 fokszámú FIR vagy IIR szűrú helyett egy 100-200-as fokszámú warpolt IIR szűrővel is ugyanazt a hatást lehet elérni. A módszernek hátulütője a magasabb számításigénye, ami a speciális szűrőstruktúra következménye.

A Kautz-szűrő a warpolt szűrő általánosításának tekinthető. A warpolt szűrővel ellentétben itt a beszúrandó mindentáteresztő szűrők nem kell, hogy azonosak legyenek, és így nem csak egy λ paraméterrel lehet hangolni a frekvenciafelbontást. A Kautz-szűrőkről bővebben Paatero és Karjalainen [2003] ír.

A harmadik, ebben a szakdolgozatban alkalmazott megoldás a párhuzamosan kapcsolt másodfokú szűrőkből létrehozott struktúra [Bank ICMC07]. A szűrő blokkvázlata 3.5-2. ábrán látható. A kimenet párhuzamosan kapcsolt másodfokú szűrők és egy opcionális FIR tag által szűrt jelek összege.



3.5-2. ábra: hangszertest párhuzamos másodfokú szűrőkkel (forrás: [Bank ICMC07])

Azért esett erre a megoldásra a választás, mert a szűrési algoritmus megegyezik a húrmodell másodfokú rezonátorainak implementációjával, valamint a szűrőtervezéshez szükséges MATLAB-függvények rendelkezésre állnak a http://home.mit.bme.hu/~bank/parfilt/ honlapon.

A levezetést Bank [ICMC07] ismerteti. Bontsuk fel a közelítendő átviteli függvényt parciális törtekre:

$$H(z^{-1}) = \sum_{k=1}^{K} c_k \frac{1}{1-p_k z^{-1}} + \sum_{m=0}^{M} b_m z^{-1}, \qquad (3.5.2)$$

ahol p_k a rendszer pólusai, melyek valós impulzusválasszal rendelkező rendszernél valósak, vagy konjugált komplex párokat alkotnak. A második tag (3.5.2) egyenletben az *M.* fokú FIR-tagot jelenti. A p_k pólusvektort a logaritmikus frekvenciafelbontás eléréséhez úgy állítsuk be, hogy a pólusok logaritmikusan helyezkedjenek el, és elég sűrűen ahhoz, hogy elég jól tudják közelíteni az eredeti

átviteli függvény pólusait (a pólusok meghatározásáról 4.4 fejezetben lesz szó). Ekkor a (3.5.2) egyenlet lineáris függvénye c_k és b_m paramétereknek, és így alkalmazhatjuk a legkisebb négyzetes eltérés módszerét azok meghatározására. A c_k és b_m paramétervektorok ismeretében (3.5.2) alapján közvetlenül megvalósítható a közelítő szűrő, párhuzamos elsőfokú szűrők segítségével. Ez azonban számítási teljesítmény szempontjából kevésbé hatékony, mint ha másodfokú szűrőket használunk.

A konjugált komplex póluspárokat közös nevezőre hozva [Bank ICMC07] mintájára:

$$\frac{c_k}{1-p_k z^{-1}} + \frac{c_{k+1}}{1-p_{k+1} z^{-1}} = \frac{c_k\left(1-p_{k+1} z^{-1}\right)+c_{k+1}\left(1-p_k z^{-1}\right)}{\left(1-p_k z^{-1}\right)\left(1-p_{k+1} z^{-1}\right)} = \frac{d_{k,0}+d_{k,1} z^{-1}}{1-\left(p_k+p_{k+1}\right) z^{-1}+p_k p_{k+1} z^{-2}}, \quad (3.5.3)$$

ahol p_k és p_{k+1}, valamint c_k és c_{k+1} konjugált komplex párok. A valós pólussal rendelkező részlettörteket elsőfokú IIR-szűrőkként, vagy kettesével másodfokú IIR-szűrőkként implementálhatjuk. A (3.5.3) egyenletben kapott forma alapján a közelítendő szűrőt párhuzamos, valós együtthatójú másodfokú szűrők, valamint az opcionális FIR-rész összegeként valósíthatjuk meg. E megoldás számításigénye sokkal kisebb, mint a párhuzamos elsőfokú szűrőké.

4 Analízis

A modellhez már csak a konkrét paraméterek szükségesek. A modális húrmodellből adódóan f_k és τ_k sorozatok leírják a húr viselkedését, így L és μ paraméterek elvesztik jelentőségüket, és végeredményben csak egy konstans szorzásnak felelnek meg. Emiatt ezeket a paramétereket nem lényeges pontos mérésből meghatározni.

A húrmodellhez szükséges paraméterek:

- μ – húr tömegsűrűsége [g/m]
- L – húr hossza [mm]
- f_k – módusfrekvenciák [Hz]
- τ_k – módusok lecsengési ideje [s]

A pengetőhöz szükséges paraméterek:

- m_h – pengető kéz tömege [g]
- m_p – pengető tömege [g]
- K_h – pengető hajlítási rugalmasságát jelentő rugóállandó [N/m]
- K_p – pengető felületi rugalmasságát jelentő rugóállandó [N/m]

A gitártest átvitelének számításához ismert gerjesztőjel és az arra adott válasz szükséges.

Az analízis MATLAB-ban történik, kihasználva a szoftver jelfeldolgozást és jelanalízist elősegítő funkcióit. Az egyes függvények megírásánál fontos szempont volt az általánosság, de mivel a végcél továbbra is egy jól működő VST plugin, a különböző gitárhúrokhoz és az eltérő minőségű mintákhoz alkalmazkodás végett a függvények finomhangolhatóak is. Ez a paraméterekben és a hibadetektálásban nyilvánul meg, az alapvető algoritmusok ugyanazok.

4.1 Mérések

Az f_k és τ_k paraméterek meghatározásához, valamint a hangszertest átviteléhez méréseket végeztünk. A szóban forgó hangszer egy Valencia CG-160 klasszikus gitár, D'Addario Classic Nylon EJ27 (Normal Tension) húrkészlettel felszerelve. A 3 vastag húr ezüstbevonatú réztekercseléssel rendelkezik, hangjuk jelentősen eltér a 3 vékony (magas) húrétól. A méréseket az I épület DSP laborjában végeztük 2013 tavaszán.

4.1.1 Pengetett hangok felvétele

Mikrofonnal felvettünk pontosan 44 gitárhangot:

- a mély E húrról ötöt, E2-G#2 tartományban,
- az A húrról ötöt, A2-C#3 tartományban,
- a D húrról ötöt, D3-F#3 tartományban,
- a G húrról négyet, G3-A#3 tartományban,
- a H húrról ötöt, H3-D#4 tartományban,
- a magas e húrról húszat, E4-H5 tartományban.

A felvételt a húrlábhoz közel végeztük, erőteljesen pengetve, hogy a magas harmonikusok is gerjesztődjenek. A mikrofont a hangnyílás fölé helyeztük, kb. 4-5 cm-re. A felvett mintákat ellenőrzés és vágás után .wav fájlokként tároltuk. Ezzel a szintetizátor terjedelme legfeljebb 44 hang lehet, E2-H5 tartományban.

Ezekből a hangmintákból később kinyerhetők a lecsengési idők és a módusfrekvenciák az egyes hangokra.

4.1.2 Gitártest impulzusválaszának mérése

Ahhoz, hogy a hangszertestet mint szűrőt implementálni tudjuk, meg kell mérnünk ismert gerjesztésre adott válaszát.

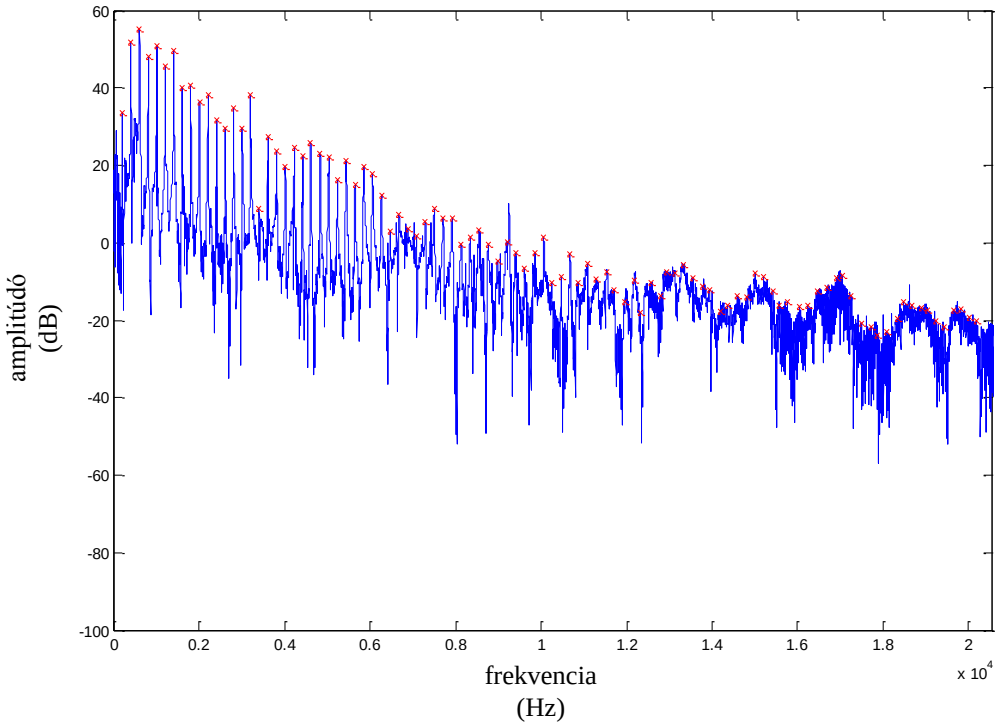
A méréshez a gitárhúrokat egy sállal letompítottuk, és egy gyorsulásmérővel felszerelt kalapáccsal megütöttük a gitártestet több helyen is (a nyeregnél, a hídnál, ezek mindkét oldalánál). A fentebb leírt elrendezésben mikrofonnal felvettük a test válaszát (ami egy koppanás), és rögzítettük a kalapács gyorsulását is. A felvett minták közül eltávolítottuk a hibás, zajos méréseket, a maradékból pedig néhány legjobbat használtuk

fel később az átvitel megállapításához. Onnan pedig azt az egy mintát alkalmaztam, amelyik a legjobb hallható eredményt adta.

4.2 Módusfrekvenciák megállapítása

A húrmodellre vonatkozó paramétereket minden egyes hangra külön-külön, a felvett mérésből állapítjuk meg. Lehetne szűkített számú mérésből is interpolálni, de mivel rendelkezésre áll az összes hang, és az analízisprogram kellően általános, nem okoz problémát minden mintát analizálni.

A módusfrekvenciák megállapításához azt a tényt használjuk ki, hogy a gitárhang spektruma vonalas: a domináns frekvenciaértékeknél az amplitúdóspektrumnak lokális maximumai vannak.



4.2-1. ábra: üresen pengetett E húr (82.4 Hz alapfrekvencia) spektruma a megtalált csúcsokkal

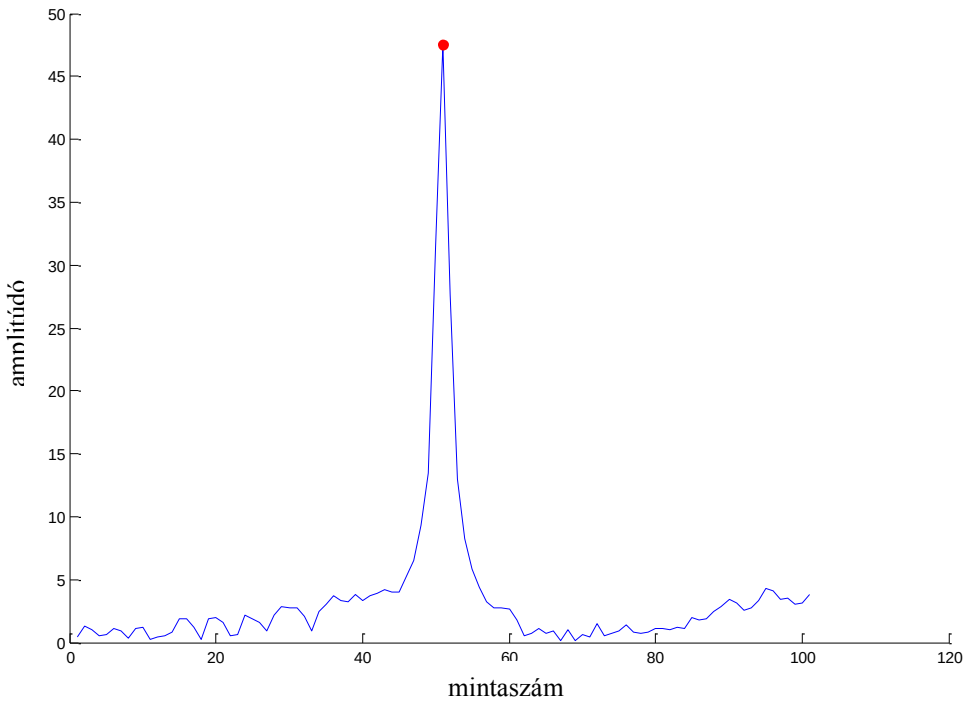
Noha a 4.2-1. ábráról ránézésre viszonylag egyszerűen megállapítható, hogy hol vannak ezek a csúcsok, algoritmizálni nem triviális a nemkívánatos zajcsúcsok miatt. Az erre szolgáló MATLAB-függvény ismertetése következik.

Bemenetként meg kell adni magát az elemzendő mintát, a vélt alapfrekvenciát, a maximális módusszámot, és a mintavételi frekvenciát. Első lépésben a mintát lerövidíti úgy, hogy 200 periódus férjen bele a vélt alapfrekvenciás módusból (ennek oka a

következő bekezdésben olvasható). Ezután a spektrum szivárgásának csökkentése céljából egy fél Hanning-ablakot helyez a jelre. Ez a frekvenciákat nem módosítja.

Ezután a spektrum számítása következik a beépített *fft* függvénnyel. Az *fft* sajátosságaiból adódik, hogy a spektrumot tároló vektor hossza megegyezik az eredeti minta hosszával, így hosszabb minta pontosabb spektrumot eredményez. Azonban ezzel több zaj is jelentkezik, ugyanis a gitárhang lecsengő volta miatt a jel-zaj viszony egyre romlik, ahogy hosszabb mintát veszünk. Az alapharmonikus 200 periódusa megfelelő hosszúságú ahhoz, hogy a pontosság ne romoljon számottevően, de elég rövid ahhoz, hogy a zaj hatását csökkentse.

A csúcskereséshez mindig a feltételezett csúcs helyének bizonyos környezetét tekintjük (4.2-2. ábra), és megkeressük az ezen a tartományon vett abszolút maximumot. Amennyiben ez nem a tartomány szélén van, úgy valószínűleg csúcsot találtunk, és a helyét kimentjük egy vektorba.



4.2-2. ábra: vizsgált tartomány és megtalált csúcs (mély E húr üresen pengetve, 1. harmonikus, kb. 82.4 Hz frekvencia)

Kezdetben a vizsgált tartomány az argumentumként megadott alapfrekvencia ±25%-os környezete, a későbbiekben pedig a következő frekvenciacsúcs körülbelüli helyét az előző értékekből interpolálja, és annak $\pm 0.5 f_0$ környezetét veszi. Mivel ideális

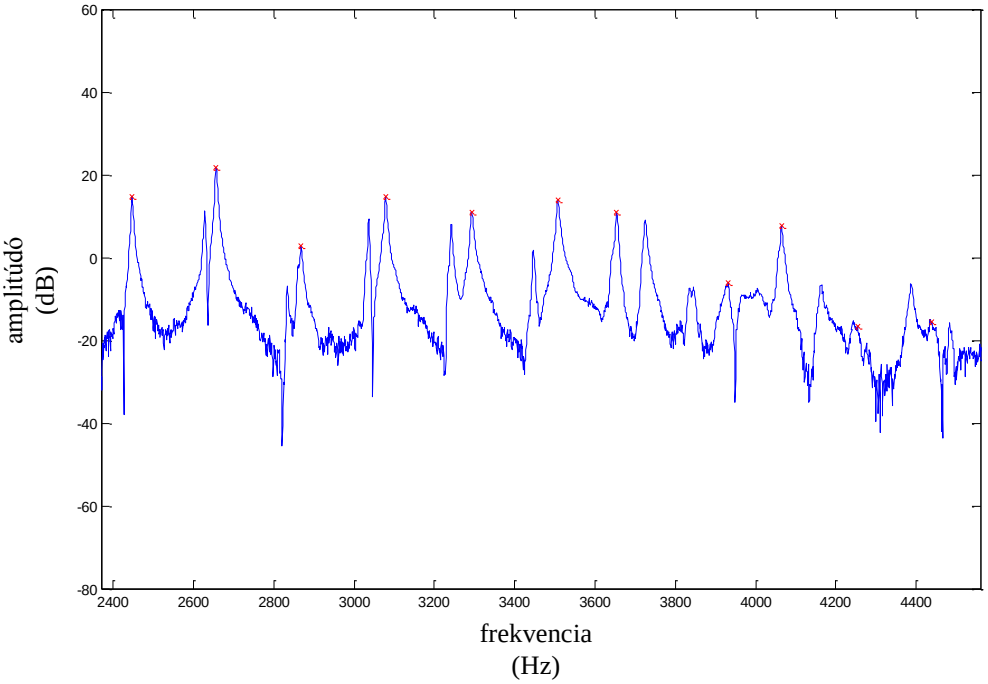
esetben a frekvenciacsúcsok távolsága pontosan f_0 lenne, $\pm 0.5 f_0$ tartomány még elég szűk ahhoz, hogy a szomszédos csúcsok hatása torzítsa számottevően a mintát, viszont elég tág ahhoz, hogy ha az adott csúcs az inharmonicitás miatt elcsúszik, akkor is megtalálja a program.

A vizsgálati tartomány kiválasztása után a beépített *findpeaks* függvény megkeresi a lokális maximumokat, és nagyság szerint csökkenő sorba rendezi őket. Innen a legelső érték helyén valószínűleg valós csúcs van. A *findpeaks* függvény előnye, hogy ténylegesen csúcsokat keres, a tartomány szélén fennálló értékeket nem veszi figyelembe. Ahhoz, hogy megkapjuk, hogy a spektrumon hányadik mintapontnál volt a csúcs, a tartományon belüli pozícióhoz hozzá kell adnunk a tartomány elejét jelentő offsetet.

Az első 10 csúcsot a legelső csúcsból származtatva keressük: az első csúcs vélt helye argumentumból adott, utána egészen a tizenegyedikig mindig az előző csúcs helyéhez f_0-t adva kapjuk a tartomány közepét.

A tizenegyedik módus után lineáris interpolációval határozzuk meg a következő csúcs helyét: az előző tíz módus helyére *polyfit* függvénnyel egyenest illesztünk, és ez kijelöli a következő frekvencia vélt helyét. Erre azért van szükség, mert a magas harmonikusok felé az megjelenik az inharmonicitás, és a módusfrekvenciák elkezdenek eltérni az ideális kf_0 értékektől.

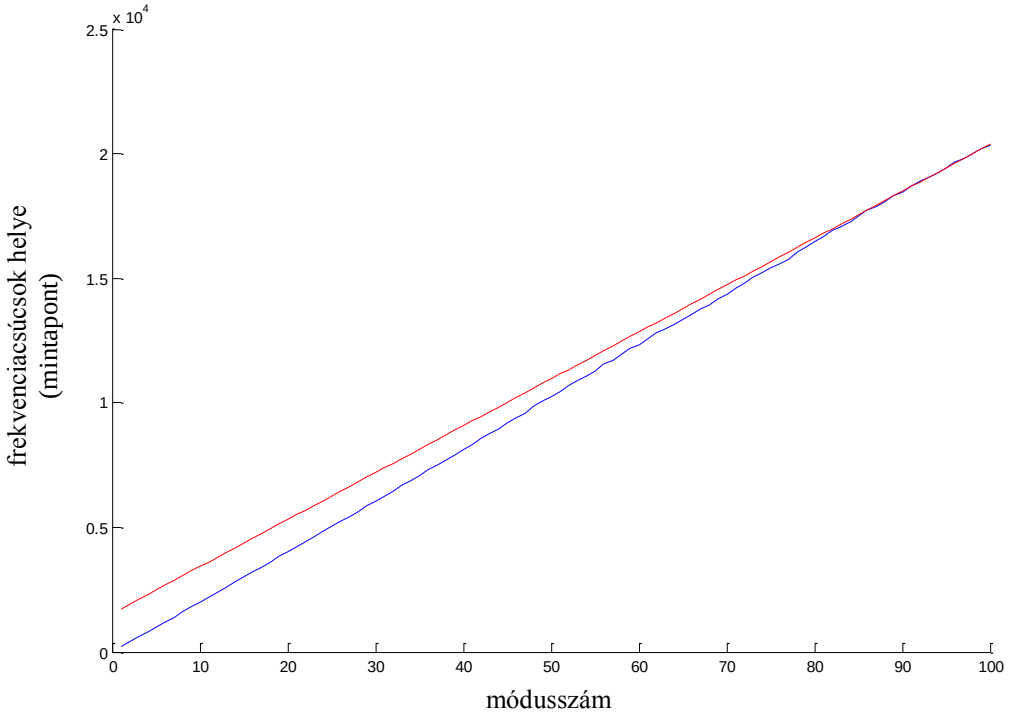
Egy másik probléma, amire ki kell térni, a lefogott hangoknál jelentkező duplacsúcsok (4.2-3. ábra). Ez az üres („open”) húroknál nem jelenik meg, bár kettős spektrumvonalak itt is előfordulhatnak a húr két polarizációja miatt. Azoknál a hangoknál azonban, amelyeknél a húr le volt fogva egy bizonyos bundnál, egymástól változó távolságra elhelyezkedő duplacsúcsok jelenhetnek meg.



4.2-3. ábra: duplacsúcsok az 1. bundnál lefogott G húron (spektrum részlete)

Ennek oka abban keresendő, hogy, míg az üres húrok a húrnyaknál rögzítve vannak, a lefogott húrokat az ujj nyomja le, hogy azok a bundra feküdjenek. A fém bundon ezután a húrnak van egy kis holtjátéka a fogólappal párhuzamos síkban fel-le, és ez azt eredményezi, hogy a fogólap síkjával párhozamos síkban a húr hosszabb szakasza tud rezegni, mint az arra merőleges síkban. Emiatt a lefogott hangoknál a két polarizáció eltérő frekvenciasorozatot eredményez, és ez a spektrumban kettős csúcsokként jelenik meg, míg a hangban lebegés-szerű effektként (erre a hatásra épít a vibrato technika). Ahhoz, hogy ezt a problémát teljesen kiküszöböljük, két frekvenciasorozatot kellene illeszteni a spektrumra, de ez túlmutat a szakdolgozat keretein.

A következő csúcsok keresésénél alkalmazott lineáris interpoláció (4.2-4. ábra) a duplacsúcsok problémáját is valamelyest orvosolja.



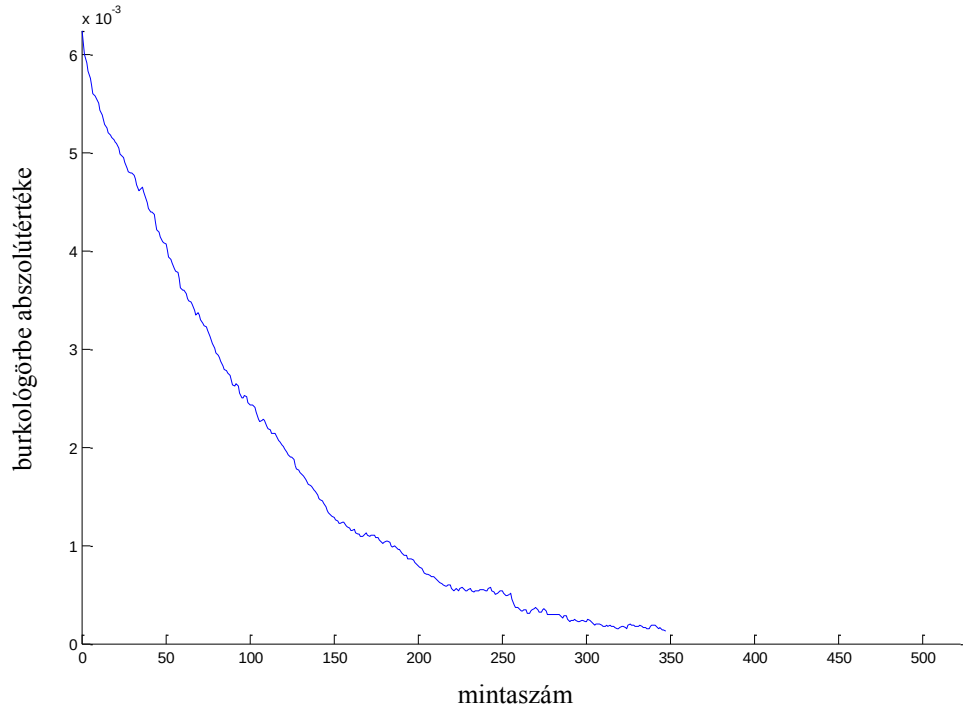
4.2-4. ábra: frekvenciacsúcsok helyei (kék) és az utóbbi 10 mintára fektetett interpoláló egyenes (piros), üresen pengetett mély E húrra

Itt meg kell még jegyeznünk, hogy a tényleges modellben nem az itt megkapott frekvenciák kerülnek felhasználásra. Ezek arra kellenek, hogy a következő pontban tárgyalt módszerhez meglegyenek a közel pontos módusfrekvenciák, amiket majd pontosítunk.

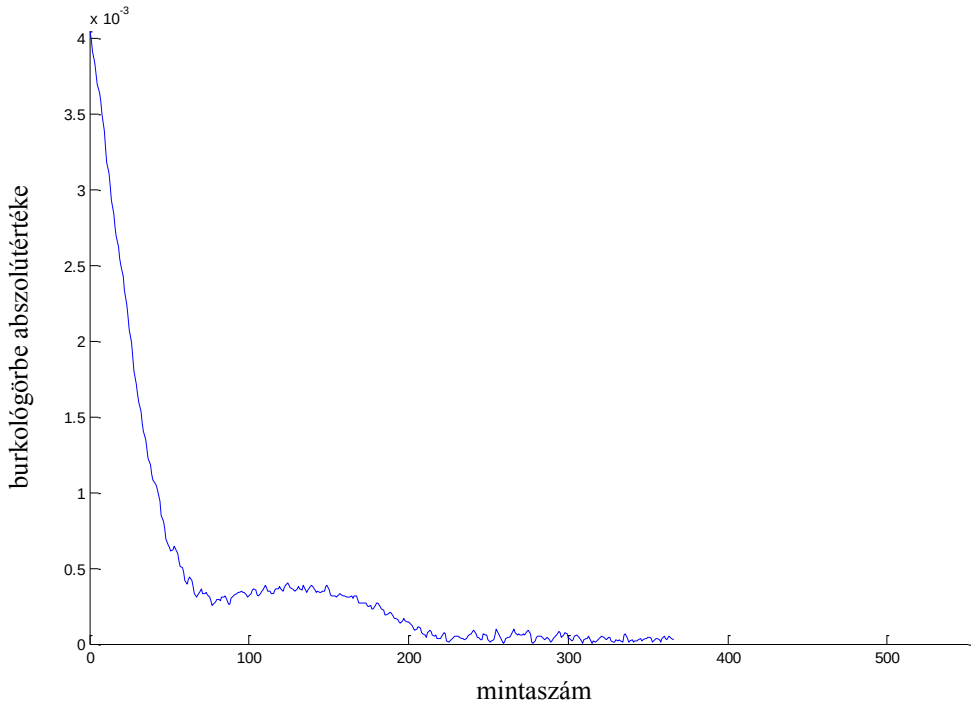
A függvény végén, amikor minden csúcs helye bekerült a kimeneti vektorba, annak minden elemét át kell skálázni $\frac{f_s}{N}$ faktorral, ahol N a bemeneti minta pontjainak száma. Erre a műveletre az *fft* már korábban tárgyalt tulajdonsága miatt van szükség.

4.3 Pontos módusfrekvenciák, lecsengési idők, amplitúdók

A lecsengési idők, a pontos frekvenciák, és az amplitúdók meghatározása az egyes módusokhoz tartozó burkológörbékre illesztett szűrők paramétereiből történik. Mindenekelőtt azonban ki kell térnünk arra a kérdésre, hogy hanyadfokú szűrőt szükséges illeszteni.



4.3-1. ábra: egy közel exponenciális lecsengés (mély E húr üresen pengetve, 5. harmonikus)

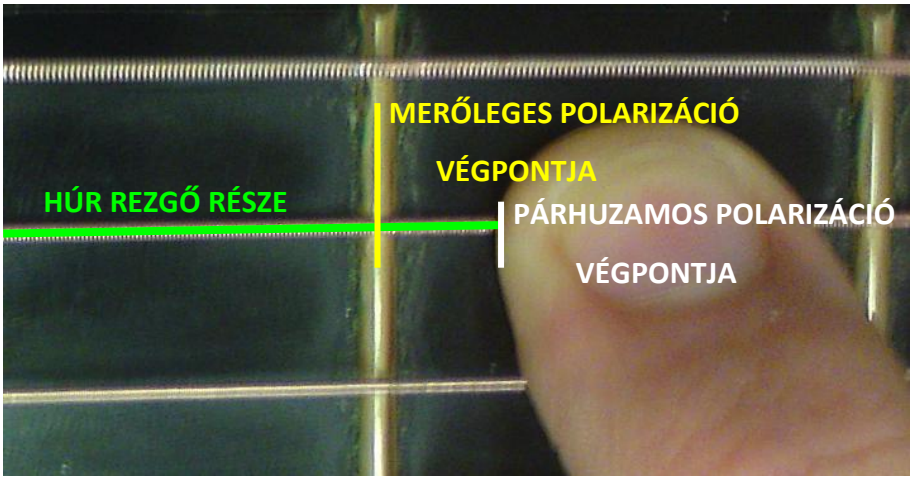


4.3-2. ábra: egy kevésbé exponenciális lecsengés (mély E húr üresen pengetve, 6. harmonikus)

A 4.3-1. ábrán látható egy olyan burkológörbe, amelyet elég jól lehetne egy darab lecsengő exponenciálissal közelíteni. Azonban a legtöbb módusnál a burkoló

inkább 4.3-2. ábrához hasonló, az előző fejezetekben leírt kettős frekvenciacsúcsok miatt. Ezt nem célszerű egy exponenciális lecsengéssel modellezni, viszont egy másodfokú komplex szűrő impulzusválasza már igen jól közelíti ezt a jelet. Ez azt jelenti, hogy minden módusra nem egy, hanem kettő frekvenciát, lecsengési időt és amplitúdót kapunk, és az ezekből alkotott két rezonátor összege adja a módust.

Az egyenleteink számításánál feltételeztük, hogy a húr egy polarizációval rendelkezhet, azaz csak egy síkban rezeg. A valóságban azonban ez nincs így, a húr két irányban tud rezegni, illetve longitudinális irányban is, de azzal a gitárszintetizátorban nem foglalkozunk.



4.3-3. ábra: párhuzamos és merőleges polarizáció rögzítési pontjai

A nem üresen pengetett („open”), hanem lefogott hangoknál még bonyolultabb a helyzet. A 4.3-3. ábrán látható a lefogott húr fogólapra merőleges és fogólapra párhuzamos polarizációjának lezárása. A fém bundon (merőleges polarizáció végpontja) a húr tud fel-le csúszkálni, így a párhuzamos polarizációban a húr hosszabbnak tekinthető, mint a merőleges polarizációban, a hosszkülönbség pedig a lefogó ujj és a bund közti távolság. Ez a fizikai modellben azt jelenti, hogy a húr valódi, térbeli rezgése két olyan rezgésből tevődik össze, amelyek módusfrekvenciái kismértékben eltérnek, mivel a két polarizációt tekintve a húr hossza nem egyforma. Ezért jelennek meg a duplacsúcsok a spektrumban, és ezért modellezünk két rezonátorral minden módust.

A gerjesztésnél azonban feltételeztük, hogy mind a pengető, mind a játékos ujja által keltett kitérés a húrt csak a fogólappal párhuzamos síkban téríti ki, ezért a húr gerjesztésre való visszahatását csak az elsődleges rezonátorokból számítoljuk. Maga a gerjesztés viszont az elsődleges és a másodlagos rezonátorra is egyformán hat.

Annak eldöntésére, hogy melyik az elsődleges és melyik a másodlagos rezonátor, azt a szabályt alkalmazzuk, hogy a másodlagos rezonátornak mindig kisebb a kezdőamplitúdója (erről ugyanebben a fejezetben később lesz szó).

Az analízisfüggvény bemenete maga a felvett hangminta, a módusfrekvenciákat analizáló csúcskereső függvény által visszaadott frekvenciavektor, valamint a mintavételi frekvencia. A minta pár másodperc hang, az elején fél-egy másodperc alapzajjal. Ez azért szükséges, hogy ez alapján meg lehessen állapítani a zajszintet a későbbi vizsgálathoz.

Az analízis a Frequency Zooming Autoregressive Moving Average (FZ-ARMA) módszerrel történik [Karjalainen et al, 2002]. Az adott módus analízisének első lépése a DC-re keverés. Tudjuk [Jury 1964]-ből, hogy a spektrum frekvenciatartománybeli $-f_{0k}$ eltolásához a jelet időtartományban szorozni kell $e^{j2\pi f_{0k}t} = e^{j2\pi\frac{f_{0k}}{f_s}n}$ taggal. Annak érdekében, hogy a vizsgált, körülbelül f_{0k} frekvencián található csúcsot lekeverjük DC-re, és a többi frekvenciacsúcsok hatásait kiküszöböljük, szorozzuk meg az eredeti jelet $e^{j2\pi\frac{f_{0k}}{f_s}n}$ taggal, majd szűrjük meg egy aluláteresztő szűrővel.

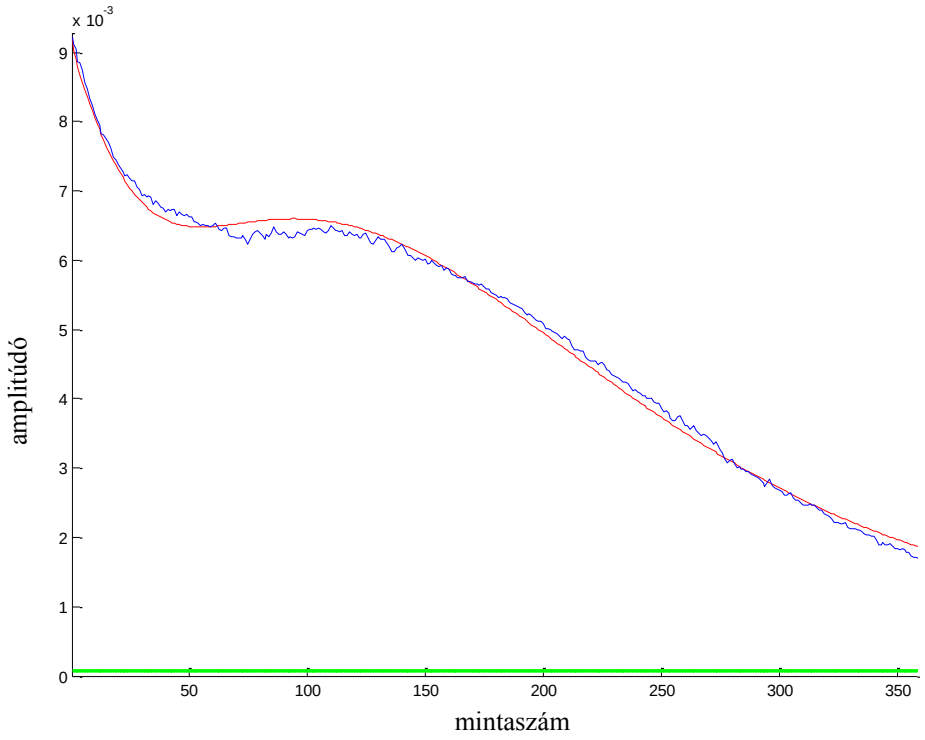
A szűrő paramétereit úgy válasszuk meg, hogy az adott, mostmár DC-re kevert frekvenciacsúcsot minél kevésbé torzítsa, de a magasabb frekvenciákat szűrje ki. Ez a probléma nem triviális, kihasználva azt, hogy a csúcsok közel f_0 távolságra vannak egymástól, valamint próbálgatással a $0.15f_0$ szélességű szűrő megfelelőnek bizonyult. Túl széles szűrő esetén a magas harmonikusok is belekerülnek a szűrt jelbe, túl szűk szűrő esetén a lecsengési idők torzulnak. A szűrő így egy $0.15f_0$ vágási frekvenciájú, másodfokú Butterworth aluláteresztő szűrő. A MATLAB *filtfilt* utasításával megszűrjük a jelet.

Ennek eredményeképp egy DC-re kevert komplex jelet kapunk, melynek amplitúdója az adott módus burkológörbéje. A következő lépés számításigényessége miatt ezt a jelet ezerszeresen decimáljuk. A tranziensek és a zaj hatása itt zavarként jelentkezik, úgyhogy a burkoló maximumhelyétől 40 mintától jobbra kezdjük a vizsgálatot, az ez előtti mintákat eldobjuk.

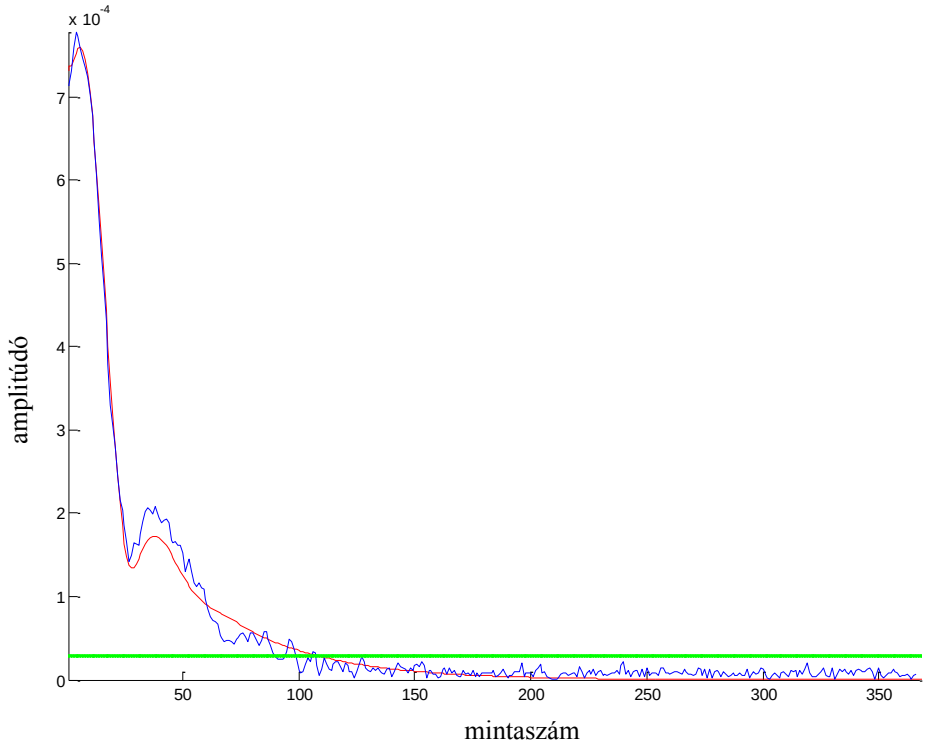
Meg kell állapítani azt is, hogy hol a minta vége, ahol már olyan rossz a jel-zaj viszony, hogy el kell dobni a mintákat. Ehhez a minta elején meghagyott alapzajt használjuk: az átlagszintjének kétszeresét (+6 dB) vesszük határértékül. Ha a jelszint

véglegesen ez alá csökken, feltehetjük, hogy onnantól a zaj a domináns, így ezeket a mintákat is eldobjuk.

Ezután jön a függvény legfontosabb eleme: szűrőtervezés az *stmcb* függvénnyel. Az identifikációt az *stmcb* függvény a Steiglitz-McBride módszer alapján végzi [Steiglitz és McBride, 1965] a legkisebb négyzetes eltérés módszerével. A függvénnyel egy két pólussal és két zérussal rendelkező komplex szűrőt illesztünk a DC-re kevert, szűrt komplex jelre. Ennek a szűrőnek az impulzusválasza komplex, és közelíti a lekevert komplex jelet. A 4.3-4. és 4.3-5. ábrákon láthatóak a lekevert komplex jelek és az illesztett szűrők impulzusválaszainak abszolút értékei. Az *stmcb* függvény a diszkrét idejű átviteli függvény számlálójának és nevezőjének együtthatóival adja meg a szűrőt. Mivel komplex szűrőről van szó, az együtthatók is komplexek.



4.3-4. ábra: eredeti jel (kék), becsült zajszint (zöld) és illesztett impulzusválasz (piros), üresen pengetett E húr, 2. harmonikus



4.3-5. ábra: eredeti jel (kék), becsült zajszint (zöld) és illesztett impulzusválasz (piros), üresen pengetett E húr, 16. harmonikus

Az *stmcb* függvényből a következő alakban kapjuk az átviteli függvényt [van den Boom 2006]:

$$H(z) = \frac{b_2 z^{-2} + b_1 z^{-1} + b_0}{a_2 z^{-2} + a_1 z^{-1} + a_0}, \qquad (4.3.1)$$

Ennek a rendszernek az impulzusválasza az eredeti burkolót közelíti, tehát az impulzusválaszból ki tudjuk nyerni a közelítő paramétereket. Az impulzusválaszt meghatározásához bontsuk fel az átviteli függvényt részlettörtekre a MATLAB *residue* függvényével, (3.5.2) egyenlet mintájára:

$$H(z) = z^{-1}\frac{r_1}{1 - p_1 z^{-1}} + z^{-1}\frac{r_2}{1 - p_2 z^{-1}} + K, \qquad (4.3.2)$$

ahol r, p és K tagok komplexek. K konstans az impulzusválasz kezdetiértékét jelenti, ez nem hordoz a modellhez szükséges információt, így elhagyjuk. A két megmaradt tag a két rezonátort jelképezi. Feltételezzük, hogy a két rezonátor frekvenciája közötti különbség néhány Hz nagyságrendű, ezt az aluláteresztő szűrő vágási frekvenciájával is korlátozzuk. A továbbiakban csak az egyik tagra ismertetem a

levezetést, a másikra ugyanígy megkapható, a Z-transzformáció disztributivitásából adódóan.

Inverz Z-transzformálva (4.3.2) első tagját, a z^{-1} késleltető tagot elhagyva [Jury 1964]:

$$Z^{-1}\left\{\frac{r_1}{1-p_1 z^{-1}}\right\} = r_1\, p_1^n\, \varepsilon[n]. \qquad (4.3.3)$$

Az $\varepsilon[n]$ egységugrást jelentő tényezőt a továbbiakban nem írjuk ki. Mivel tudjuk, hogy r_1 és p_1 paraméterek komplexek:

$$r_1\, p_1^n = |r_1|e^{j\varphi(r_1)}\,|p_1|^n e^{j\varphi(p_1)n} \qquad (4.3.4)$$

Tudjuk, hogy ezzel egy lecsengő szinuszos függvényt modellezünk, amelyet frekvenciatartományban eltoltunk egy komplexszel történő szorzással. A (3.3.6b) egyenletet módosítsuk a fázis figyelmen kívül hagyásával, és írjuk fel a függvényt:

$$y[n] = A_1 e^{-\frac{n}{\tau_1 f_s}} \cos\left(2\pi\frac{f_1}{f_s}n + \varphi_1\right) e^{j\,2\pi f_{01}\frac{n}{f_s}} \qquad (4.3.5)$$

Megjegyzendő, hogy ebben az esetben az f_1, τ_1 jelölések nem az első módus paramétereit jelentik, hanem a *k*-adik módus két rezonátora közül az elsőt, továbbá f_s a decimálás utáni mintavételi frekvencia. A φ_1 taggal eddig nem foglalkoztunk és a jövőben sem fogunk, de a matematikai modell pontosságához szükséges. Ugyanezen oknál fogva használhatunk *cos* függvényt, annak érdekében, hogy a későbbiekben egyszerűsödjenek az egyenletek.

A komplex számok tulajdonságaiból tudjuk, de egyszerűen le is vezethető, hogy $\cos(\varphi) = \frac{e^{j\varphi}+e^{-j\varphi}}{2}$. Ezt felhasználva (4.3.5) egyenleten:

$$y[n] = \frac{A_1}{2} e^{-\frac{n}{\tau_1 f_s}} \mathrm{e}^{\mathrm{j}(2\pi\frac{f_1}{f_s}n+\varphi_1)}\, e^{j\,2\pi f_{01}\frac{n}{f_s}} + \frac{A_1}{2} e^{-\frac{n}{\tau_1 f_s}} \mathrm{e}^{-\mathrm{j}(2\pi\frac{f_1}{f_s}n+\varphi_1)}\, e^{j\,2\pi f_{01}\frac{n}{f_s}}$$

Összevonások után a két tag körfrekvenciája:

$$\omega_1 = 2\pi\frac{1}{f_s}(f_1 + f_{01}), \qquad \omega_2 = 2\pi\frac{1}{f_s}(-f_1 + f_{01}).$$

Feltételezésünk szerint f_1 közel esik f_{01} frekvenciához, tehát az első tag frekvenciája közel $2f_{01}$, míg a másodiké közel 0. A lekeverés után elvégzett aluláteresztő szűrés tehát az első tagot gyakorlatilag teljesen elnyomta, a másodikat pedig közel torzítatlanul hagyta. Tehát ami marad:

$$y[n] = \frac{A_1}{2} e^{-\frac{n}{\tau_1 f_s}} e^{j\left(2\pi\frac{n}{f_s}(f_{01}-f_1)-\varphi_1\right)}. \qquad (4.3.6)$$

Ezt az alakot már egyenlővé tehetjük (4.3.4) egyenlettel:

$$\frac{A_1}{2} e^{-\frac{n}{\tau_1 f_s}} e^{j\left(2\pi\frac{n}{f_s}(f_{01}-f_1)-\varphi_1\right)} = |r_1|\,|p_1|^n e^{j(\varphi(p_1)n+\varphi(r_1))}$$

Tudjuk, hogy két komplex szám akkor és csak akkor egyenlőek, ha az abszolút értékük és fázisuk is egyenlő. Tegyük egyenlővé az abszolút értékeket:

$$\frac{A_1}{2} e^{-\frac{n}{\tau_1 f_s}} = |r_1|\,|p_1|^n$$

Elkülönítve a konstans és az időfüggő tagokat és átrendezve:

$$A_1 = 2|r_1|, \qquad (4.3.7a)$$

$$\tau_1 = \frac{-1}{f_s \ln(|p_1|)}. \qquad (4.3.7b)$$

Egyenlővé téve a fázisokat:

$$2\pi\frac{n}{f_s}(f_{01}-f_1) - \varphi_1 = \varphi(p_1)n + \varphi(r_1)$$

Elkülönítve a konstans és az időfüggő tagokat és átrendezve:

$$f_1 = \frac{-f_s\,\varphi(p_1)}{2\pi} + f_{01}, \qquad (4.3.7c)$$

$$\varphi_1 = -\varphi(r_1). \qquad (4.3.7d)$$

A (4.3.7) egyenletek tehát megadják mindhárom paramétert, amit kerestünk, továbbá a fázist is, ami nem szerepel a húrmodellben.

Figyelembe kell még venni, hogy, mivel a szűrőt nem a burkoló maximumhelyétől illesztettük, hanem attól még 40 mintával későbbtől, a (4.3.7a) egyenletben kapott paraméterből még vissza kell számolni a 40 mintával ezelőtti értéket:

$$A_1 = \frac{2|r_1|}{e^{-\frac{40}{f_s \tau_1}}} \qquad (4.3.7e)$$

A levezetés pontosan ugyanez a második rezonátor paramétereire is. A Z-transzformáció tulajdonságaiból [Jury 1964] tudjuk, hogy két tag összegének Z-transzformáltja a tagok Z-transzformáltjának összegével egyezik. Ezért a (4.3.2)

egyenlet második tagjával is ugyanezt a műveletsort kell elvégezni, hogy megkapjuk a módus második rezonátorának paramétereit.

Annak eldöntésére, hogy melyik legyen az elsődleges, és melyik a másodlagos rezonátor, azt a szabályt alkalmazzuk, hogy az elsődleges rezonátor amplitúdója nagyobb. Azért ebből indulunk ki, mert a másodlagos rezonátor csak mint kiegészítő rezonátor szerepel, a hang alapvető tulajdonságait a főrezonátorok adják.

A modellben A_1 paramétert a (3.3.3a) egyenletből kapjuk, így erre nincsen szükség. Viszont a hangzásban fontos paraméter az elsődleges és másodlagos rezonátorok amplitúdóinak aránya, tehát $A_r = A_2/A_1$. Ezt az értéket azonban a 4.5 fejezetben leírt módon, a hibajavítással egy függvényben számoljuk, ezért az analízisfüggvénynek mind A_1, mind A_2 kimenete.

Ezzel megkaptunk minden paramétert, ami a húrmodellhez szükséges. Az értékek valódiságát a hibajavító függvény ellenőrzi, amelyről a 4.5 fejezetben lesz részletesen szó.

4.4 Gitártest átvitele

A 4.1.2 fejezetben tárgyalt módon megmértük a gitártest ismert gerjesztésre adott válaszát. Ebből a két mintából fel tudjuk írni a hangszertest átviteli függvényét és az impulzusválaszát, amelyekből frekvenciatartománybeli szorzással illetve konvolúcióval megkaphatjuk a gitártest kimeneti jelét bármilyen bemenetre.

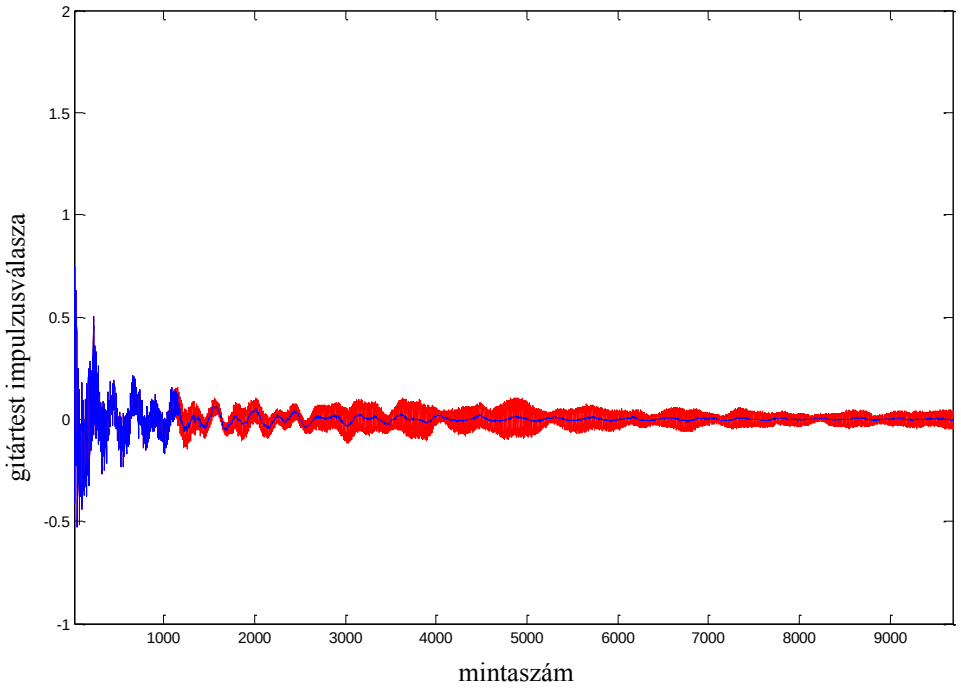
Az átvitel megállapításának legegyszerűbb módja, ha veszünk a gerjesztésből és a válaszból is azonos számú mintát, *fft* függvénnyel Fourier-transzformáljuk, majd elemenként elosztjuk a válasz transzformáltjának vektorát a gerjesztés transzformáltjának vektorával. A kimeneti vektor maga az átviteli függvény, annak *ifft*-vel kapott inverz Fourier-transzformáltja az impulzusválasz. Ezt a választ még minimálfázisúvá kell alakítani a MATLAB *rceps* függvényével, annak érdekében, hogy a párhuzamos szűrőstruktúrában a FIR-tagokat ne kelljen implementálni (lásd 3.5. fejezet), hogy megkapjuk a használható impulzusválaszt.

A helyzet azonban nem ennyire egyszerű a mérésnél jelen lévő alapzaj, valamint az egyébként nem hallható magasfrekvenciás zajok miatt. Ezek hatásának kiküszöbölésére a következő műveleteket végezzük el a számított impulzusválaszon:

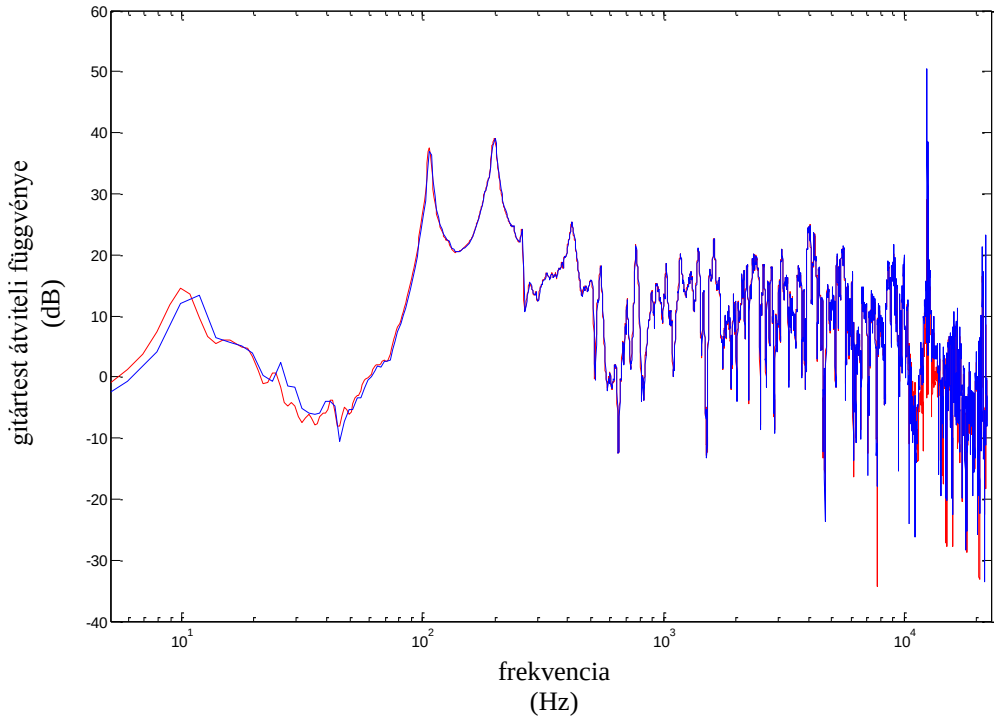
1. frekvenciatartományban alul-, és felüláteresztő szűréssel szétbontjuk alacsony-, és magasfrekvenciás összetevőre
2. mindkét összetevőnél külön-külön megnézzük, hogy időtartományban mikor tűnik el a jel, és mikortól marad már csak zaj
3. mindkét összetevőt megszorozzuk egy olyan szélességű Hanning-ablakkal, hogy a jel lényeges részét ne torzítsuk, de a végén lévő zajt elnyomjuk
4. a két összetevőt újra összeadva megkapjuk a javított impulzusválaszt.

Az első lépésben az alul-, és a felüláteresztő szűrő vágási frekvenciája egyaránt $0.25\frac{f_s}{2}$. Ez a használt $f_s = 44100\,Hz$ mintavételi frekvencia esetén $f_c = 5512.5\,Hz$. Ezt a döntést indokolja, hogy a húrmodell által előállított hangok domináns módusainak frekvenciái e vágási frekvencia alá esnek: 110 *Hz*-es A hang esetén az első 50, 440 *Hz*-es A hang esetén az első 12 módus ebbe a tartományba esik.

A szűrés után kapott két jelre ezután a kirajzolt grafikonok alapján megállapítjuk, hogy mikortól simulnak bele a környezeti zajba, és egy-egy olyan függvénnyel ablakozzuk őket, ami a jel hasznos részénél 1 értékű, majd egy fél Hanning-ablakkal átmegy 0 értékbe, elnyomva ezzel a jel végén fellépő alapzajt. A fél Hanning-ablak szélessége 100 minta, ami *0.002* másodpercnyi átmenetnek felel meg. A két lekevert jelet összeadva visszakapjuk a teljes impulzusválaszt (4.4-1. ábra), de az alapzajt és a jel végén fellépő egyéb zavarokat kiszűrtük. Frekvenciatartományban a változás a 4.4-2. ábrán látható.



4.4-1. ábra: eredeti impulzusválasz (piros), és javított minimálfázisú impulzusválasz (kék)



4.4-2. ábra: gitártest átviteli függvénye az impulzusválasz javítása és minimálfázisúvá alakítása előtt (kék) és után (piros)

A 3.5 fejezetben leírt módon megadott pólusokkal ezután másodfokú szűrőt terveztetünk. Ehhez nagy segítséget nyújtanak a http://home.mit.bme.hu/~bank/parfilt

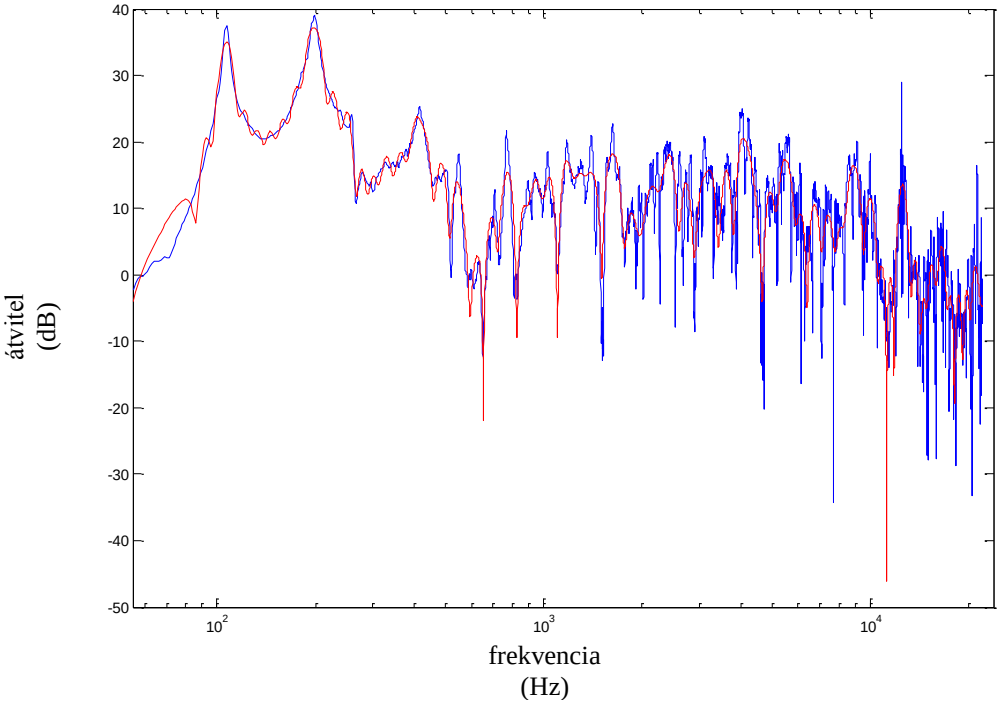
oldalon található MATLAB-függvények, amelyek a megadott pólusfrekvenciákból pólusokat, azokból pedig a másodfokú szűrők együtthatóit állítják elő.

A logaritmikus elhelyezkedésű pólusfrekvenciák meghatározásához figyelembe vesszük, hogy a húrmodell által kiadott jel frekvenciája sosem kisebb, mint 50 *Hz*, valamint hogy a 20 *kHz* fölötti frekvenciákat már nem halljuk. Az előállított frekvenciavektor elemei $bázisfrekvencia \cdot frekvenciafelbontás^{i}$ egyenletből jönnek, ahol ebben az esetben *bázisfrekvencia = 50 Hz,* i értéke *1* és *N* közé esik, a *frekvenciafelbontás* értékét pedig addig állítjuk, amíg az átviteli függvény már megfelelően közelíti a mért átvitelt. A párhuzamos szűrők számát jelentő *N* értéket a maximális pólusfrekvenciából (20 kHz) és a frekvenciafelbontásból számoljuk. A végső modellben a frekvenciafelbontás *1.2, N=33* párhuzamos szűrővel. Ebből a frekvenciavektorból a *freqpoles* függvény állít elő pólusokat.

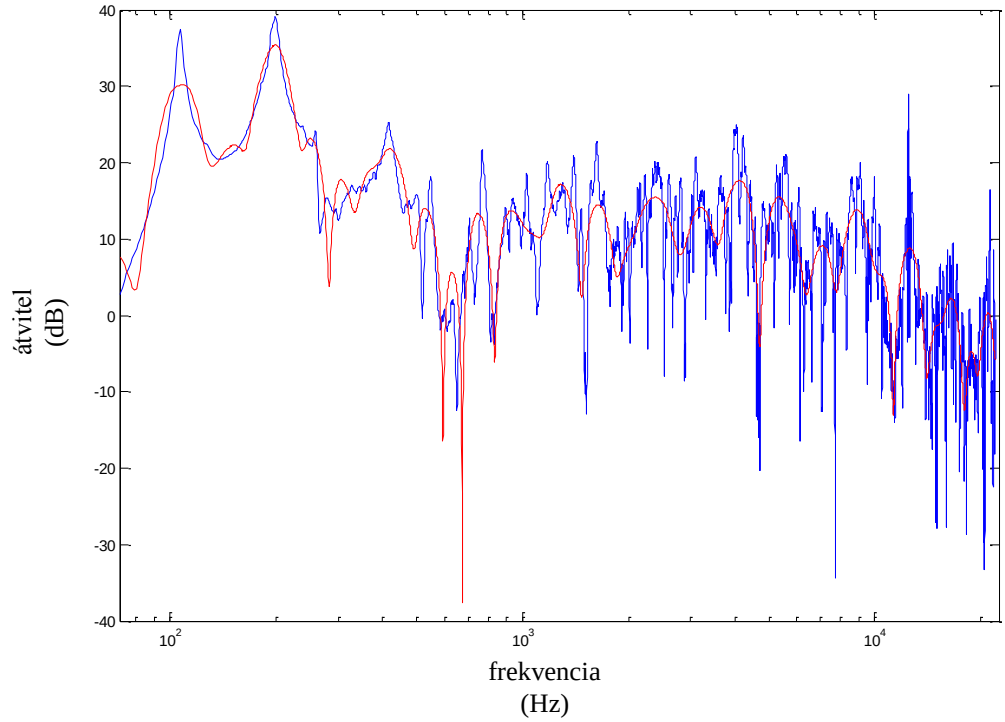
Az így előálló pólusokat és a javított impulzusválaszt kapja meg *parfiltdes* függvény, amely előállítja a párhuzamos szűrők pólusait és zérusait, *Am* (3x*N*) és *Bm* (2x*N*) méretű mátrixok formájában, ahol *N* a logaritmikus elhelyezkedésű pólusok száma. A függvény *NFIR* paraméterét, amely a párhuzamos FIR-tagok számát jelenti, nullára állítjuk, mert a FIR-részt nem használjuk a modellben.

A szűrő paramétereiből ábrázolhatjuk annak átviteli függvényét, és összevethetjük az eredeti átvitellel. A 4.4-3. ábrán 63 párhuzamos szűrővel, *1.1* frekvenciafelbontással megvalósított szűrő átvitele látható, míg a 4.4-4. ábrán 33 szűrővel, *1.2* frekvenciafelbontással előállított szűrőé. A finomabb felbontás láthatóan jobb eredmény ad, de *1.1* frekvenciafelbontással 63 párhuzamos szűrőt kellene implementálni, ami túlzottan számításigényes lenne. További értékek vizsgálata után jó kompromisszumnak bizonyult az *1.2* felbontású szűrő (4.4-4. ábra), amely hallhatóan jó eredményt ad, de még nem terheli a processzort túlzottan, így ez került be a végleges modellbe.

A szűrőtervezés után a párhuzamos szűrők számát, valamint együtthatóikat tartalmazó *Am* és *Bm* mátrixokat kiírjuk a *guitarbody.txt* nevű szöveges fájlba, ahonnan a VST plugin később beolvassa azokat.



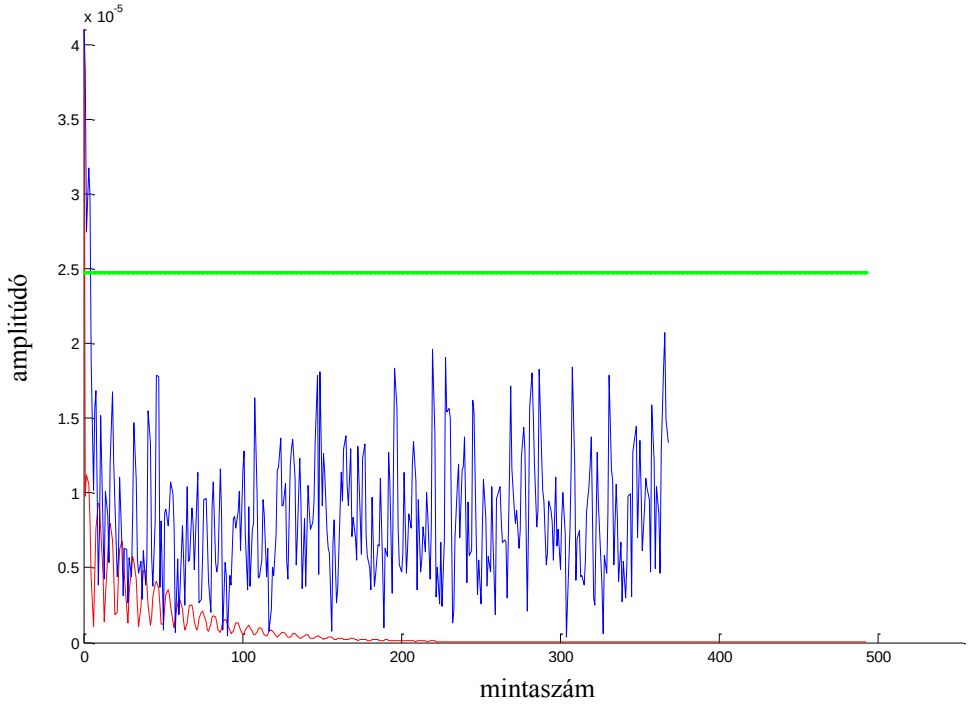
4.4-3. ábra: eredeti (kék), és párhuzamos szűrőkkel megvalósított átviteli függvény (piros), 63 párhuzamos szűrővel



4.4-4. ábra: eredeti (kék), és párhuzamos szűrőkkel megvalósított átviteli függvény (piros), 33 párhuzamos szűrővel

4.5 Hibajavítás

A módusfrekvenciák keresésénél elkerülhetetlenül előfordul, hogy zajcsúcsot jelcsúcsként értelmez az analízisprogram. A burkolóillesztő függvény azonban ezen a frekvencián csak zajt talál, és emiatt az illesztett impulzusválasz és a paraméterek is hibásak lesznek. A 4.5-1. ábrán látható egy hibásan analizált, zajos minta.



4.5-1. ábra: zajos minta és a ráillesztett impulzusválasz (üresen pengetett E húr, 17. harmonikus)

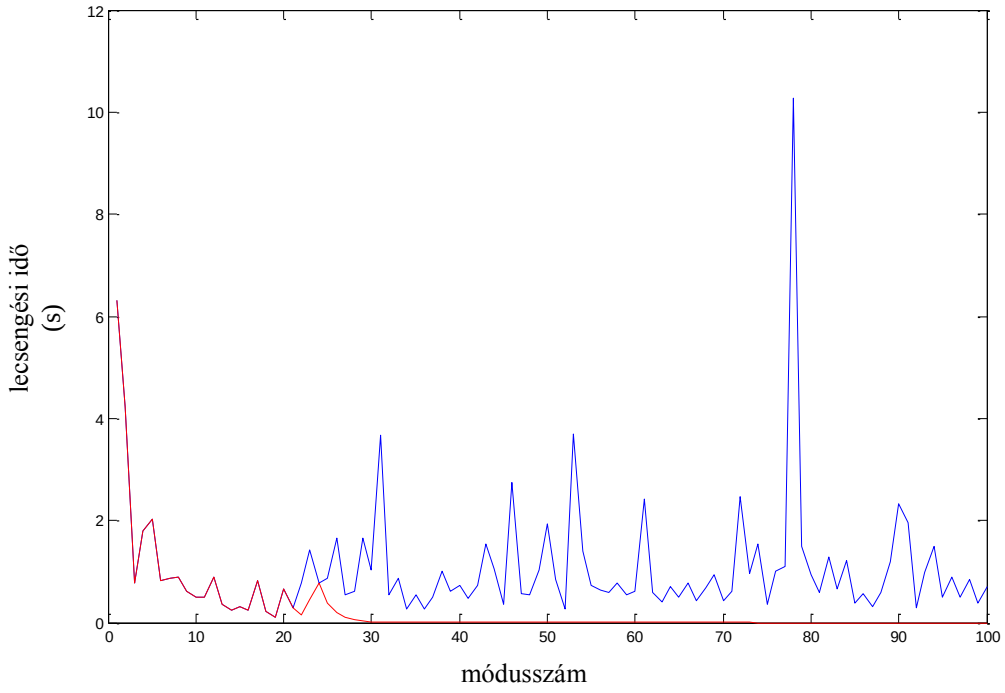
Bizonyos esetekben nyilvánvaló, hogy hiba történt (például negatív lecsengési idők esetén), más esetekben az értékek ésszerű tartományba esnek, de a kimenetben hallható hibát okoznak. Ezért ezeket a hibás értékeket meg kell keresni, ki kell venni, és a helyükre valamilyen módszerrel új értékeket kell meghatározni.

Az erre szolgáló függvény bemenetei a paramétervektorok: $f_{1,k}$, $f_{2,k}, \tau_{1,k}, \tau_{2,k}$, $A_{1,k}$, $A_{2,k}$, valamint a vizsgált módusszám.

Az első vizsgált paraméter a $\tau_{1,k}, \tau_{2,k}$ lecsengési idők. Ezek a paraméterek közvetlenül megjelennek a modellben, ezért a kiugró, hibás értékek a kimenetben is észlelhető hibát okoznak. Feltételezhetjük továbbá, hogy ha egy lecsengési idő rossz, akkor az ahhoz a módushoz tartozó amplitúdó is az. A két rezonátorbank paraméterit külön vizsgálja a program, tehát a főrezonátorban egy hibás érték javítása nem változtatja meg a másodlagos rezonátor adott módushoz tartozó értékeit, és fordítva.

A lecsengési idők vizsgálatánál azt a feltételezést vesszük alapul, hogy a lecsengési idő a magasabb módusok felé haladva általában csökken, továbbá hogy a negatív érték biztosan hiba. A függvény végigfut mindkét rezonátor lecsengési idejein, és ha a *k*-adik lecsengési idő magasabb, mint az előző *ablak* számú érték maximuma az adott lecsengésiidő-vektorban, vagy negatív, akkor hibásnak jelöli, és nullára állítja az adott módushoz tartozó paramétereket. Az *ablak* paraméter hangolható, és az adott mintához kézzel be kell állítani, értéke 5-25 között szokásos.

Ezzel kiszűrtük a hibás értékeket, azzal, hogy nullára állítottuk őket. Így viszont bizonyos módusok kiesnek, amitől szegényebbé válik a hang. Ennek orvoslására a nullázott értékek helyére lineáris interpolációval állítunk elő új értékeket: minden módusra a hibásnak jelölt paraméterek helyére az eggyel kisebb és eggyel nagyobb számú módus paraméterének számtani közepét szúrjuk be. A 4.5-2. ábrán látható egy hibás lecsengésiidő-vektor, és annak javított változata.



4.5-2. ábra: eredeti (kék) és javított (piros) lecsengési idők (üresen pengetett E húr)

Már csak egyetlen hibalehetőség maradt: az amplitúdók. Noha $A_{1,k}$ paraméter nem jut be a modellbe, hiszen ott (3.3.3a) szerint számolunk amplitúdókat, közvetetten mégis. $A_{2,k}$-t ugyanis az eredeti minta alapján számoljuk, mégpedig úgy, hogy az $A_r = \frac{A_2}{A_1}$ arányt használjuk a modellben, és a (3.3.3a) alapján számolt A_1-et A_r

paraméterrel megszorozva kapjuk A_2-t. Ezért problémát okoz, hogyha egy adott módusnál az eredeti A_1 túl közel esik A_2 értékhez, mert ez magas A_r-t eredményez. Ez a hiba az implementációban úgy jelenik meg, hogy egy darab, jellemzően gyors lecsengésű harmonikus lesz domináns.

Ezt kiküszöbölendő a következő szabály érvényes: ha egy módus második rezonátorának lecsengési ideje kevesebb, mint 0.1 másodperc, akkor azt a második rezonátort eldobjuk. Ezzel a nagyon kis lecsengési idejű, többnyire nem is hallható módusokkal nem kell foglalkozni.

4.6 MATLAB implementáció

A MATLAB-ban történő analízis után egy szintetizátor-prototípust implementáltam szintén MATLAB-ban. Ennek előnyös tulajdonsága a VST pluginhoz képest, hogy könnyen nyomon követhetőek a folyamatok, felrajzolhatók a jelalakok. Hogyha a prototípus már jól működik, a szintetizátort át lehet portolni C++ nyelvre, és beleépíteni a VST keretbe (5. fejezet).

A húrmodellben előállított maximális módusszámot *KMAX=100* értéknek választottam. A 4.5-2. ábrán is látható, hogy az ilyen magas harmonikusok már elhanyagolhatóak a kimeneti hang szempontjából, viszont a processzort terhelik, és ezzel rontják a hatékonyságot. A többi hangra is megvizsgálva a grafikonokat és a kiadott jelet az látszik, hogy legfeljebb 100 módusba minden hang lényeges része belefér, így ennél többet felesleges implementálni. Az egyes hangokhoz tartozó tényleges módusszám ennél kisebb is lehet.

A pengetés implementálása a 3.4 fejezetben leírt egyenletek alapján történt, a húrlábra ható erő (a húrmodell kimeneti jele) pedig 3.3 fejezetben leírtak szerint. A gitártestet a 3.5 fejezet alapján párhuzamos másodfokú szűrőkkel valósítottam meg. E másodfokú szűrők bemenetére érkezik a húrmodell kimeneti jele, és az egyes szűrők kimeneteinek összege adja a gitártest által szűrt jelet.

A (3.3.8) egyenletekből látható, hogy a szintézishez közvetlenül csak b_k, $a_{1,k}$, és $a_{2,k}$ paraméterek szükségesek. Az analízisprogram azonban $f_{1,k}$, $f_{2,k}$, $\tau_{1,k}$, $\tau_{2,k}$ és A_r paramétereket szolgáltat, valamint $A_{1,k}$ vektort számítjuk a módusfrekvenciák és a statikus paraméterek alapján. Ahhoz, hogy ezeket a paramétereket át tudjuk alakítani a szintézishez szükséges állandókká a (3.3.8) egyenletek szerint, további számítási

kapacitásra van szükség. Ezért ezt a pluginban nem valós időben számoljuk, hanem a MATLAB alatt írt prototípus végén egy fájlba exportáljuk, ahonnan a VST plugin hangonként kiolvassa azokat. A fájl neve *„NOTE.txt”*, ahol *NOTE* a hang MIDI kódját jelöli, és a legmélyebb hanghoz (üresen pengetett E húr) a 40-es kód tartozik, tehát ebben az esetben a fájl *„40.txt”* lesz. A fájl tartalma a következőképpen épül fel:

K	$a_{11,\mathrm{k}}$	$a_{12,\mathrm{k}}$	$a_{21,\mathrm{k}}$	$a_{22,\mathrm{k}}$	$b_{1,\mathrm{k}}$	$b_{2,k}$

Az egyes elemek egy darab szóközzel vannak elválasztva, jelentésük:

- K a megvalósítandó módusszám, a következő vektorok elemszáma
- $a_{11,\mathrm{k}}$ az elsődleges rezonátor $a_{1,\mathrm{k}}$ paramétervektora, az egyes elemek szóközzel elválasztva
- $a_{12,\mathrm{k}}$ a másodlagos rezonátor $a_{1,\mathrm{k}}$ paramétervektora
- $a_{21,\mathrm{k}}$ az elsődleges rezonátor $a_{2,\mathrm{k}}$ paramétervektora
- $a_{22,\mathrm{k}}$ a másodlagos rezonátor $a_{2,\mathrm{k}}$ paramétervektora
- $b_{1,\mathrm{k}}$ az elsődleges rezonátor b_{k} paramétervektora
- $b_{1,\mathrm{k}}$ a másodlagos rezonátor b_{k} paramétervektora

Ezeket a paramétereket az 5. fejezetben leírt módon az egyes hangokhoz tartozó szintetizátorok konstruktorai kiolvassák a megfelelő fájlokból.

A gitártest paramétereit a 4.4 fejezetben leírtak szerint számoljuk, és a MATLAB-prototípus ezeket is kiírja egy fájlba, ahonnan a plugin beolvashatja a szűréshez.

5 VST implementáció

5.1 Bevezetés

Annak érdekében, hogy a gitárszintetizátort virtuális hangszerként lehessen használni erre alkalmas szoftverekben (pl. Cubase, Ableton, FL Studio, stb.), VST pluginként implementáltam, C++ nyelven.

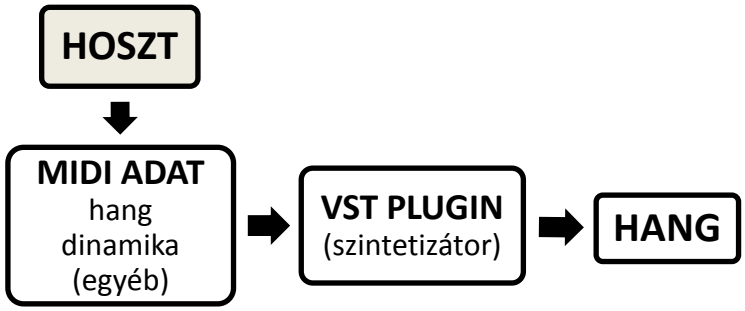
Ezek a szoftverek úgynevezett digitális audio munkaállomások, virtuális hangstúdiók. Segítségükkel hangeffektek, hangok, zeneszámok állíthatók elő, illetve ezeket effektezni, módosítani is lehet. Akár effektről, akár virtuális szintetizátorról beszélünk, ez mindig egy olyan alszoftvert (plugint) jelent, amelynek bemenete és kimenete is audio adat, és a főszoftver (a hoszt) hívja őket. Effektek esetén a bemenet és a kimenet is valódi hang, míg szintetizátorok esetén a bemenet egy MIDI struktúra.

A MIDI formátum digitális hangszerek vezérlőparamétereinek tárolására szolgál. Tartalmazza, hogy milyen hangszerről van szó, az adott hang hangmagasságát, a hang dinamikáját, és egyéb tulajdonságokat. A számítógépre köthető MIDI billentyűzetek is ilyen adatot állítanak elő. Ezek az eszközök digitális zongorára hasonlítanak, de sok esetben nem tudnak önmaguktól hangot kiadni, hanem USB vagy MIDI csatlakozón keresztül számítógépre küldik a leütött hangok adatait, MIDI formátumban. Ezután a számítógépen futó hoszt program feldolgozza a MIDI adatot, és megszólaltatja a beállított virtuális hangszert (például ebben az esetben a gitárszintetizátort). A virtuális hangszert egy *.dll* kiterjesztésű plugin fájl tárolja, amely VST-szabvány szerint íródott.

A VST (Virtual Studio Technology) egy szoftveres interfész, amelyet a Steinberg cég fejlesztett ki. A C vagy C++ nyelven, a Steinberg által biztosított keretben írt *.dll* formátumú pluginok a hoszt program és a tényleges effekt vagy szintetizátor között biztosítják a kommunikációt.

Alapvetően háromféle VST plugin létezik. Az effekt pluginok a hoszt által küldött hangot dolgozzák fel, azt effektezik valamilyen algoritmus alapján, vagy kijelzik és információt szolgáltatnak róla. A MIDI effekt pluginok MIDI üzeneteket dolgoznak fel, és továbbítják őket más pluginok felé.

A VST pluginok harmadik típusa a virtuális hangszerek csoportja. Ezek többnyire virtuális szintetizátorok vagy samplerek. Egy ilyen plugint a hoszt MIDI üzenetekkel vezérel, amelyek leírják a be-, vagy kikapcsolandó hang paramétereit. Én egy ilyen szintetizátort készítettem, amelynek magját ugyanaz az algoritmus képezi, amit korábban MATLAB-ban implementáltam prototípusként. A hoszttal való kommunikációhoz és bizonyos funkciók megvalósításához további függvények megírására volt szükség C++ nyelven. A MIDI esemény feldolgozását mutatja be leegyszerűsítve az 5.1-1. ábra.



5.1-1. ábra: VST szintetizátor folyamatábrája

5.2 Szintetizátor objektum

A polifónia megvalósítása érdekében minden egyes hangot előállító szintetizátort egy külön objektumpéldányban kell megvalósítani, majd a hanggenerálási szakaszban a bekapcsolt hangok megfelelő függvényét hívni (lásd 5.2.2 fejezet).

5.2.1 Tagváltozók

Minden ilyen objektumpéldány tartalmazza az adott szintetizátor működéséhez szükséges összes információt: a húr rezonátorainak együtthatóit, a pengetéshez szükséges erő-, és pozícióadatokat, valamint a húrra és a pengetőre jellemző globális konstansokat is. Ezen kívül minden példány tartalmazza a kívülről változtatható paramétereket: a hangerőt, a pengetés dinamikáját, a pengetés módját (ujj vagy pengető), valamint a pengetés helyét. Azt az információt, hogy az adott szintetizátor aktív-e, a *SynthIsOn* bool típusú változó tárolja, míg azt, hogy meg van-e már pendítve a húr, *peng*, szintén bool típusú változó.

5.2.2 Tagfüggvények

A legelső hívott függvény az objektum konstruktora. Ennek első lépése a 4.6 fejezetben leírt módon, MATLAB-ban előállított szövegfájl megnyitása. A szóközzel elválasztott, meghatározott sorrendben tárolt paramétereket ezután a megfelelő tagváltozóba tölti. Inicializálja a változtatható paramétereket egy előre megadott értékre. A jel előállításához szükséges erő-, és pozícióértékeket, valamint a módusalakok aktuális, előző, és előző előtti értékeit egy külön függvény, a *resetSynth()* hívásával állítja a kezdeti értékükre.

A *resetSynth()* függvényben elvégzett műveletekre a hang kikapcsolásánál is szükség van, ezért kerültek külön tagfüggvénybe. A kezdeti és kiindulási értékeket kinulláza, kikapcsolja a szintetizátort (*SynthIsOn = false).*

A hang kikapcsolását jelentő MIDI esemény érkezésekor az objektumpéldány *noteOff()* függvénye hívódik, ami mindössze a *resetSynth()* függvényt hívja, azonban a későbbi bővíthetőség és általánosság miatt szükség van mindkét tagfüggvényre.

Ha olyan MIDI üzenet érkezik, amely a hang bekapcsolását kéri, adott *velocity* dinamikával (amelynek értéke 0 és 127 között változhat) és *dFrames* mintával a buffer kezdőpointeréhez képest, a *noteOn(int velocity, int dFrames)* függvény hívódik. Ez a *velocity* alapján beállítja azt az *Fh_hatar* erőt, ahol a pengető elhagyja a húrt, és megszólal a hang. A legnagyobb velocityhez az a legnagyobb erő tartozik, ahol ez még éppen megtörténik (*Fh_hatar = 0.6*). A *currentDelta* tagváltozóba eltárolja *dFrames* értékét, és *SynthIsOn* változó *true*-ra állításával jelzi, hogy a hang aktív.

Az objektum központi tagfüggvénye, amely maga a szintetizátor, a *generateOutSample(float* out, float sampleRate)* függvény. Amennyiben az adott hang aktív, ez a függvény hívódik az aktuális mintára, melynek helyét *out* pointer adja meg. A pointer által mutatott értékhez adja hozzá a függvény a saját maga által generált kimenetet.

A függvény első lépésben megnézi, hogy *currentDelta* értéke nagyobb-e, mint 0. Ha igen, az azt jelenti, hogy az aktuális mintában ennek a szintetizátornak még nem kell megszólalnia. Ekkor *currentDelta* csökkentése után visszatér.

Ha *currentDelta* nem nagyobb, mint 0, megnézi, hogy *peng* változó értéke logikai hamis-e. Ez a változó azt tárolja, hogy meg lett-e már pendítve a húr. Ha nem, először le kell futtatni a hang megszólalása előtti műveleteket, amely beállítja a kezdeti

értékeket. Ezek a húr és a pengető között ható erő, a húr kitérése, és a sebességek számítása, miközben a módusamplidútók beállnak a kezdeti értékükre. Amikor a húr és a pengető közötti erő eléri a dinamika által meghatározott *Fh_hatar* értéket, a ciklus befejeződik, és amennyiben pengetővel pengettük a húrt, *peng* értéke igazra állítódik.

Ez a ciklus akkor is lefut, ha ujjal pengettünk, viszont akkor még egy művelet hiányzik, mielőtt *peng* értéke igaz lesz. A ciklusban előállított *Ys* érték jelenti a pengetés pillanatában a húr kitérését a pengetési pontban. Ez az *Ys* szorzóként jelenik meg (3.4.2) egyenletben, amellyel az így adott háromszög magasságát állítja be.

Amennyiben *peng* már igaz volt a függvény kezdetén (mert a kezdeti értékek már korábban beálltak), vagy épp most állítottuk be a kezdeti értékeket, következik maga a hangszintézis. Ez ugyanaz az algoritmus, mint a MATLAB-ban megírt prototípus. A kimeneti pointer által mutatott értékhez módusonként hozzáadódik az adott mintához tartozó kimeneti érték.

Az egyes hangokhoz tartozó szintetizátorokat a *VstXSynth* osztály hívja. Amikor minden bekapcsolt szintetizátor elvégezte a kimenetének előállítását, a teljes kimenetre a *VstXSynth* osztály meghívja a *filter* tagfüggvényét, amely a gitártestet valósítja meg (lásd következő fejezet).

5.3 Kommunikációs és segédfüggvények

A VST hoszt szoftver MIDI üzenetek formájában küldi az adatot a plugin felé. Ezek lekezelésére egy külön *VstXSynth* osztályt kellett készíteni, amely az *AudioEffectX* származtatott osztálya.

Az osztály tagváltozói az összes szintetizátort érintő, változtatható paraméterek (hangerő, pengetési mód, pengetés helye), a szintetizátorok tömbje, és azok száma, a gitártestet modellező szűrő együtthatói, valamint az egy időben engedélyezett szintetizátorok aktuális és maximális száma. Ezek szerint adott számú szintetizátort tárolunk egy tömbben, és ezeket külön-külön vezéreljük. A hoszt program *VstXSynth* osztállyal kommunikál közvetlenül. A plugin importálásakor hívódik az objektum konstruktora, amely beállítja a szintetizátorok számát 44-re, létrehozza a szintetizátortömb elemeit, és beolvassa a *guitarbody.txt* fájlból a MATLAB-függvény által kiírt paramétereket, amelyek a gitártestet modellező párhuzamos szűrők száma, és azok együtthatói.

Az osztály két központi függvénye a *processEvents*, amely a hoszt által küldött MIDI vezérlőüzenetek feldolgozását végzi, és a *processReplacing*, amely a kimenetet állítja elő.

5.3.1 A *processEvents* függvény

A MIDI üzenetek feldolgozását a *processEvents* függvény végzi. Az alábbi táblázatból kiolvashatók a MIDI event struktúrában található *MIDIData[]* tömb lehetséges elemei:

Státuszbájt D7----D0	**Adatbájt(ok) D7----D0**	**Leírás**
1000nnnn	0kkkkkkk 0vvvvvvv	Note Off esemény: az adott hang kikapcsolását jelenti. (kkkkkkk) a hang azonosítója, (vvvvvvv) a *velocity* érték.
1001nnnn	0kkkkkkk 0vvvvvvv	Note On esemény: az adott hang bekapcsolását (megszólaltatását) jelenti. (kkkkkkk) a hang azonosítója, (vvvvvvv) a *velocity* érték.
1011nnnn	0ccccccc 0vvvvvvv	All Notes Off esemény: minden hang kikapcsolását jelenti, ha *c* értéke 123 és 127 közé esik, és *v=0*.

5.3.1. táblázat: MIDI üzenet felépítése (forrás: http://www.midi.org/techspecs/midimessages.php)

Ezeken kívül másféle üzenet is érkezhet, de csak a fentiek relevánsak a szintetizátorok működése szempontjából. Egy üzenet tehát egy státuszbájtból és két adatbájtból áll.

Amennyiben „note off” esemény érkezik, vagy olyan „note on”, ahol a *velocity* érték nulla (némely hoszt programok így kezelik a „note off” eseményt), a *processEvents* függvény megvizsgálja, hogy az adott hanghoz (*note*) tartozó szintetizátor állapota *SynthIsOn == true*, és ha igen, meghívja a szintetizátor *noteOff* függvényét, valamint az aktuálisan bekapcsolt szintetizátorok számát tároló *currentSynths* értékét eggyel csökkenti..

Ha „note on” esemény érkezik *velocity>0* értékkel a *note* hangra, a függvény először megvizsgálja, hogy az adott pillanatban a bekapcsolt szintetizátorok száma eléri-e a megadott maximumot. Ha nem, a *note* hanghoz tartozó szintetizátor *noteOn(velocity, deltaFrames)* függvénye hívódik, és *currentSynths* növekszik eggyel, ellenkező esetben pedig nem történik semmi, a hang nem szólal meg.

A harmadik típusú MIDI üzenet, a *status == 0xb0* értékű „all notes off” azt jelenti, hogy minden szintetizátort ki kell kapcsolni. Ekkor egy ciklussal végigmegy a szintetizátortömb összes tagján, és amelyik *on* állapotban van, kikapcsolja a *noteOff* tagfüggvénnyel, és a *currentSynths* tagváltozót nullázza.

A *noteOn* függvény paramétereként szereplő *deltaFrames* egész szám, és azt az értéket jelenti, amennyi mintával később kell csak megszólaltatni a hangot. Erre azért van szükség, mert a hoszt mindig *sampleFrames* számú mintát kér be a plugintól, így ha nem a legelső mintától kell kezdeni a hanggenerálást, megad egy nullától különböző *deltaFrames* értéket.

5.3.2 A *processReplacing* függvény

Magáért a hangszintézisért a *processReplacing(float** inputs, float** outputs, int sampleFrames)* függvény felel. Ez paraméterül kapja a bemeneti minták helyét (*float** inputs*, erre a szintetizátornál nincsen szükség, effektek számára van fenntartva), a kimeneti minták bufferét (*float** outputs*), valamint hogy hány mintát kell kitölteni (*int sampleFrames*). A plugin sztereó hangot állít elő, de mindkét csatornára ugyanazt a jelet teszi ki. Így végeredményben alapvetően monó a hang, de effektezéskor és egyéb műveletek elvégzésekor a két csatornát külön lehet kezelni. A hanggenerálást a függvény a bal csatornára végzi el (*out1* pointer által kijelölt memóriaterület), majd a ciklus végén a jobb csatornába is beleírja a kimeneti értéket (az *out2* pointer által kijelölt memóriaterületre).

A fő ciklus *sampleFrames* alkalommal fut le, és első lépésben kinullázza *out1* által mutatott kimeneti minta értékét, majd a szintetizátortömb összes elemén végigfut. Vizsgálja, hogy az adott szintetizátor be van-e kapcsolva (amit egy megfelelő MIDI event eredményez), és ha igen, meghívja annak *generateOutSample* tagfüggvényét *out1* kimeneti bufferre. Ezek a tagfüggvények hozzáadják a generált kimenetüket a mutató által mutatott értékhez, így mikor az összes szintetizátor elvégezte a dolgát, a kimenet polifonikus lesz, és az összes megszólaltatott hangot tartalmazza. Ezt a kimenetet kell még megszűrni a gitártesttel.

A szűrést a *filter* tagfüggvény végzi. A párhuzamos szűrők paramétereit a konstruktor beállította. Mivel ezek a szűrők is másodfokúak, a szűrési algoritmus hasonlít a húrmodellben lévő hanggenerálási ciklushoz: végigfut az összes párhuzamos szűrőn, mindegyikkel előállítja a kimenetet ugyanabból a bemeneti mintából, és a

kimeneteket összegzi. Természetesen ehhez el kell tárolni a húrmodell jelének előző értékét, továbbá minden egyes párhuzamos másodfokú szűrő előző és kettővel ezelőtti kimeneti értékét.

Végül *out1* és *out2* mutatókat lépteti eggyel (a következő mintára), és eggyel csökkenti a hátralévő *sampleFrames* értéket.

A plugin tehát a polifóniát úgy valósítja meg, hogy minden hang szintetizátorát külön kezeli, és a *note* hangot tartalmazó *noteOn* MIDI-esemény hatására bekapcsolja az *note* hanghoz tartozúó szintetizátort. Annak érdekében azonban, hogy a plugin túl sok hang egyszerre történő megszólaltatásával ne terhelje túl a processzort, az egyidejűleg aktív szintetizátorok számát korlátozni kell. Mivel egy hathúros gitáron legfeljebb 6 hang szólhat egyszerre, a *VstXSynth* osztály *maxSynths* tagváltozójának értéke 6. Így ha a *currentSynths* változó értéke eléri a 6-ot (amely az aktuálisan bekapcsolt állapotú szintetizátorok számát jelenti), a további *noteOn* üzeneteket figyelmen kívül hagyja a program.

5.4 Paraméterek és felhasználói felület

A pluginnek három kívülről állítható paramétere van, és ezeket a paraméterek minden szintetizátornál egyszerre állítódnak. Ezek a hangerő, a pengetési mód (ujjal vagy pengetővel), és a pengetés helye a húron. Az 5.4-1. ábrán látható a plugin vezérlőfelülete, FL Studio alatt.

5.4-1. ábra: FL Studio szoftver felhasználói felülete a gitárszintetizátorhoz

A felhasználó felületen található paraméterek kijelzését és állítását végző függvényeket kötött szintaktika szerint kell beprogramozni. Ezeket a Steinberg által szolgáltatott VST keretből alakítottam át.

Az egyes szintetizátorok paramétereinek állításához egy ciklus végigfut az összes objektumpéldányon, és beállítja az *fVolume, fPickingStyle, fPosition* értékeket az új értékre. Ezt meg lehetett volna valósítani *static* változókkal is, ebben az esetben nem kellene minden példányt külön-külön állítani, de esetleges későbbi bővíthetőség érdekében inkább példányonként tárolom a paramétereket.

Ezzel a VST plugin minden lényeges részét tárgyaltuk. A fő szintézisciklus a MATLAB-prototípus átportolása C++ nyelvre, a felhasználói felülethez kapcsolódó függvények a Steinberg által szolgáltatott üres mintaprojektből vannak. Az egyetlen egyedi rész a polifóniát és a vezérlést megvalósító, 5.3 fejezetben leírt osztály.

A plugint alapvetően FL Studio 10 és MiniHost szoftverekkel teszteltem, de mivel a szabvány általános, bármilyen DAW (Digital Audio Workstation) szoftverrel működik.

5.5 Optimalizáció

Audio alkalmazásoknál nagyon fontos, hogy a program jó hatékonysággal fusson. A processzor túlterhelése esetén a minták generálásában késések jelentkeznek, és a hang nem lesz folyamatos, amitől az egész plugin használhatatlanná válik. Ezt szem előtt tartva a szintetizátor függvényeinek megírásakor olyan megoldásokat alkalmaztam, amelyek a hatékonyságot növelik.

A szintetizátor osztály *generateOutSample* tagfüggvényében, amely a kimenetet generálja, a módusokon végighaladó ciklus nem egyesével, hanem tízesével generálja a módusokhoz tartozó kimeneti jelet. Ez azt jelenti, hogy a ciklusmagban egyszerre tíz módust kezel, így a módusszámnak tízzel oszthatónak kell lennie. Ennek érdekében az eredeti módusszámból csak annyit veszünk, hogy tízzel osztható legyen. Ez többnyire megegyezik az eredeti módusszámmal, mert már a MATLAB-ban írt analízisfüggvények is 100 módussal számolnak, de magasabb hangoknál nem fér bele ennyi a spektrumba. Ilyenkor lefelé tízesre kerekít a *generateOutSample* függvény. A ciklusnak ilyen módú változtatása növeli a hatékonyságot, mert a ciklus a belépési pontján kiüríti a pipeline-t. Így ez nem minden, hanem csak minden tízedik módus számítása után történik meg.

Ott, ahol több konstans vagy ritkán változó paraméterrel gyakran kell műveletek végezni, a paramétereket összevonhatjuk egy darab konstanssá, és ezeket a konstansokat előre kiszámítva eltárolhatjuk. Ezzel a memóriaigény nő, de a számítási igény csökken. Ilyen megoldás található például a *generateOutSample* függvény fő ciklusában, amikor a módusamplidútókból húrlábra ható erőt számolunk:

$$F_{b,k}[n] = \frac{T_0\pi}{L} k\left(y_{k,1}[n] + y_{k,2}[n]\right), \qquad (5.5.1)$$

amely a (3.3.5) egyenlet implementációja a szintetizátorban. A $\frac{T_0\pi}{L}k$ tényezőket előre kiszámítjuk, és eltároljuk egy vektorban, ezután a lebegőpontos szorzásműveletek helyett egy memóriahozzáféréssel kapjuk a szorzótényezőt.

A rezonátorok zérusával is elvégezzük ugyanezt. A (3.3.2e) egyenlet szerint számítjuk a húr x_0 pontjára ható erőt:

$$F_{y,k}[\mathrm{n}] = \sin\left(\frac{k\pi x_0}{L}\right)\mathrm{u}[\mathrm{n}], \qquad (5.5.2)$$

majd a (3.3.9) egyenletből a kimeneti jelet:

$$y_k[n] = b_k \sin\left(\frac{k\pi x_0}{L}\right)\mathrm{u}[\mathrm{n}-1] - a_{1,k}y_k[n-1] - a_{2,k}y_k[n-2]. \qquad (5.5.3)$$

Az első tagban $b_k \sin\left(\frac{k\pi x_0}{L}\right)$ szorzótényezőket szintén előre kiszámíthatjuk, és egy vektorban tárolva a számításigényes *sin* műveletet elhagyhatjuk a valósidejű implementiácóból.

Azokban a ciklusokban, amelyekben erre lehetőség van, *for* és nem nulla értékhez hasonlító *while* ciklus helyett kihasználjuk a processzorarchitektúrák azon tulajdonságát, hogy az „*==0*” műveletet egy utasításból képesek elvégezni, míg nem nullához hasonlításnál legalább két-három számítási-, és memóriaműveletre van szükség. Ezért a *„for (int i=0; i<sampleFrames; i++”* típusú megoldások helyett ahol csak lehetett, *„while (sampleFrames--)”* jellegű, nullához hasonlító ciklusfeltételeket alkalmaztam.

Ezekkel a módosításokkal a program számításigénye észrevehetően csökkent, miközben a memóriaigénye továbbra is 3 kB alatt maradt, ami elhanyagolható a hoszt program memóriaigényéhez képest is. Végeredményben a párhuzamos szűrőket is implementálva egy Intel Core i3 típusú processzor egyszerre 3-4 hangot tud késés nélkül generálni, míg egy Intel Core i5 processzor számára a maximális 6 hangot egyidejű megszólaltatása sem okoz gondot.

6 Összefoglalás, fejlesztési lehetőségek

A szakdolgozat célja egy hatékonyan működő, valósághű hangú, fizikai alapokra felépített gitárszintetizátor elkészítése volt. Ezt a célt sikerült elérni. A hangszer felépítése, az egyes elemek működése mind fizikai egyenletekre épül. A VST plugin hangolható paraméterekkel rendelkezik, és az E2-H5 tartományban képes gitárhang előállítására.

A virtuális gitárszintetizátor készítése során többször is egyszerűsítésekhez kellett folyamodni, mert bizonyos problémák megoldása túlzottan időigényes vagy bonyolult lett volna. Ettől függetlenül a végeredmény egy kielégítő minőségű virtuális szintetizátor, amely betölti a funkcióját, testreszabható, és fizikai alapokra épül. Azonban még így is sok lehetőség maradt a fejlesztésre.

Az analízis részben mindenképpen nagy javulást eredményezne, ha egy frekvenciasorozat helyett kettőt illesztetnénk a spektrumra. Ez sokkal pontosabban modellezné a lefogott hangoknak azt a tulajdonságát, hogy a fogólappal párhuzamos síkban a húr kicsivel hosszabbnak tekinthető, mivel a bundon fel-le tud csúszkálni.

Pontosabb zajdetekcióval kevesebb hibás (zajos) burkolót kellene vizsgálni, és optimális esetben a hibajavító függvényt akár el is lehetne hagyni.

A mérések tisztaságát nagyban javítaná, ha valódi süketszobában megismételnénk azokat. Sajnos a DSP Laboratóriumban nem voltak ideálisak a körülmények: a légkondicionáló rendszer és a számítógépek hangja, a külső zajok, valamint kismértékben a visszhangok is rontották a mérések minőségét. Ezeket kiküszöbölve lényegesen jobb eredményt lehetne elérni.

A plugin hatékonyságát különféle C és C++ nyelvi „trükkök” segítségével lehetne javítani. Ilyenek már most is találhatók a kódban, de a mögöttes architektúra részletesebb ismeretével még gyorsabbá lehetne tenni a szintetizátort, ami több párhuzamos hang megszólaltatását tenné lehetővé, és régebbi, kisebb számítási teljesítményű PC-ken is gond nélkül futna. A legjobb teljesítmény elérése érdekében közvetlenül assembly nyelven írt kódrészleteket lehetne beszúrni, ez viszont már jóval túlmutat a szakdolgozat keretein.

Összességében a kitűzött célokat sikerült megvalósítani: a szakdolgozat kézzelfogható végterméke egy olyan VST plugin, amelyet rendeltetésének megfelelően lehet használni, fizikai alapokon, elméleti megfontolásokon és méréseken alapul, és megfelelő minőségben képes a klasszikus gitár hangját visszaadni.

7 Irodalomjegyzék

Arfib, D. (1978). Digital Synthesis of Complex Spectra by Means of Multiplication of Non Linear Distorted Sine Waves, *AES Convention 59 (February 1978), Paper Number:1319*
http://www.aes.org/e-lib/browse.cfm?elib=3035

Bank, B. (2000). Physics-Based Sound Synthesis of the Piano, *Master's thesis, Helsinki University of Technology, 2000*
http://home.mit.bme.hu/~bank/thesis/pianomod.pdf

Bank, B. (2006). Physics-based Sound Synthesis of String Instruments Including Geometric Nonlinearities, *Ph.D. thesis, Budapest University of Technology and Economics, 2006*
http://home.mit.bme.hu/~bank/phd/phd.pdf

Bank, B. (2010). A Modal-Based Real-Time Piano Synthesizer, *IEEE TRANS. ON Audio,Speech, and Language Processing, Vol. 18, No. 4, PP. 809–821*
http://home.mit.bme.hu/~bank/publist/taslp10.pdf

Bank, B. (ICMC07). Direct Design of Parallel Second-order Filters for Instrument Body Modeling, *Helsinki University of Technology*
http://home.mit.bme.hu/~bank/publist/icmc07.pdf

Van den Boom, T. (2006). Discrete-time systems analysis, *Additional Lecture Notes for the course SC4090*
http://www.dcsc.tudelft.nl/~sc4060/transp/discreteNOTES.pdf

Chaigne, A. and Askenfelt, A. (1994). Numerical simulations of piano strings I. A physical model for a struck string using finite difference methods, *J. Acoust. Soc. Am. 95, 1112 (1994)*
http://eaton.math.rpi.edu/csums/papers/instruments/Numerical%20simulations%20of%20struck%20strings.%20I.pdf

Chowning, J. (1973). The Synthesis of Complex Audio Spectra by Means of Frequency Modulation, *Computer Music Journal, Vol. 1., No. 2., April 1977, PP. 46-54* http://people.ece.cornell.edu/land/courses/ece4760/Math/GCC644/FM_synth/Chowning.pdf

Hiller, L. and Ruiz, P. (1971a). Synthesizing musical sounds by solving the wave equation for vibrating objects: Part 1 *J. Audio Eng. Soc. 19(6): 462–470.*

Hiller, L. and Ruiz, P. (1971b). Synthesizing musical sounds by solving the wave equation for vibrating objects: Part 2 *J. Audio Eng. Soc. 19(7): 542–550.*

Jury, E. (1964). Theory and Application of the z-Transform Method *Wiley, 1964*

Karjalainen, M. and Smith, J. (1996). Body modeling techniques for string instrument synthesis *Proc. Int. Computer Music Conf., PP. 232-239.*

Karjalainen, M. et al. (2002). Optimalization techniques for parametric modeling of acoustic systems and materials, *Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing, Finland, 2002.* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.3927&rep=rep1&type=pdf

Le Brun, M. (1979). Digital waveshaping synthesis, *JAES Volume 27 Issue 4 PP. 250-266.* http://www.aes.org/e-lib/browse.cfm?elib=3212

Morse, P. M. and Ingard, K. (1968). Theoretical Acoustics, *Princeton University Press, 1987*

Paatero, T. and Karjalainen, M. (2003). Kautz Filters and Generalized Frequency Resolution: Theory and Audio Applications, *JAES Volume 51 Issue 1/2 pp. 27-44* http://www.aes.org/journal/online/JAES_V51/1_2/

Perng, C. Y. J. et al. (2010). Sound synthesis of the harpsichord pluck using a physical plectrum-string interaction model, *J. Acoust. Soc. Am. 128, Issue 4* http://dafx10.iem.at/papers/PerngSmithRossing_DAFx10_P93.pdf

Rabiner, L. and Gold, B. (1975). Theory and application of digital signal processing, *Prentice-Hall, Inc., 1975*

Roads, C. (1995). The Computer Music Tutorial, *The MIT Press, Cambridge, Massachusetts, USA, 1996*

Smith, J. (1983). Techniques for Digital Filter Design and System Identification with Application to the Violin, *PhD thesis, Stanford University, California, USA, 1983* *https://ccrma.stanford.edu/files/papers/stanm14.pdf*

Smith, J. (1991). Viewpoints on the history of digital synthesis, *ICMC-91, Montreal, PP. 1-10* https://ccrma.stanford.edu/~serafin/UVA/MUSI445/hw/kna.pdf

Smith, J. (1992). Physical modeling using digital waveguides, *Computer Music Journal, Vol. 16, No. 4, PP. 74-91.* https://ccrma.stanford.edu/~jos/pmudw/pmudw.pdf

Smith, J. (2005). Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects, *CCRMA, Department of Music, Stanford University, Stanford, California* https://ccrma.stanford.edu/~jos/pasp/

Smith, J. and Serra, X. (1990). Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition, *Computer Music Journal, Vol. 14, No. 4, PP. 12-24.* http://www.jstor.org/stable/3680788

Steiglitz, K. and McBride, L. E. (1965). A Technique for the Identification of Linear Systems, *IEEE Transactions on Automatic Control, Vol.10 , Issue 4, PP. 461 - 464* http://www.cs.princeton.edu/~ken/stmcb.pdf

Verma, T. S. and Meng, T. H. Y. (1995). An analysis/synthesis tool for transient signals that allows a flexible sines+ transients+ noise model for audio, *Department of Electrical Engineering, Stanford University, Stanford, CA, USA* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.1332&rep=rep1&type=pdf

Wikipedia 1. Harmonic partials on strings, http://en.wikipedia.org/wiki/File:Harmonic_partials_on_strings.svg

Zygmund, A. (2002). Trigonometric Series, Volume 1. *Cambridge University Press, 2002*