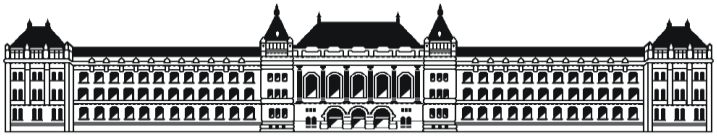


Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

SZAKDOLGOZAT

Virtuális analóg szintetizátor megvalósítása

Szalontai Péter

konzulens: dr. Bank Balázs

Budapest, 2009

Hallgatói nyilatkozat

Alulírott Szalontai Péter, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója 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.

Tudomásul veszem, hogy az elkészült szakdolgozatban található eredményeket a Budapesti Műszaki és Gazdaságtudományi Egyetem, a feladatot kiíró egyetemi intézmény saját céljaira felhasználhatja.

Kelt: Budapest, 2009. 12. 11.

…….

Szalontai Péter

Tartalomjegyzék

Kivonat

Napjainkra a számítógép a zeneszerzés, zeneszerkesztés elsődleges platformjává vált. Különösen az amatőr zenészek körében, akiknek tömegei számára a költséges és nagy helyigényű stúdióberendezések nem elérhetőek. Egy PC, egy közepes minőségű hangkártya és néhány szoftver együttese e széles rétegek számára kielégítő megoldást jelent. Egyre növekvő igény mutatkozik a zeneszerzést támogató szoftveres alkalmazásokra. Ilyen alkalmazások a virtuális hangszerek is, melyek közül sok egy-egy létező hangszer szimulációja.

Szakdolgozatomban egy virtuális analóg szintetizátor megvalósítási kérdéseivel foglalkozom. Az analóg szintetizátorok leggyakoribb szintézis elve a szubtraktív szintézis, mely a hangot egy nagy felharmonikus tartalmú jel szűrésével állítja elő. A dolgozat központi témája a szubtraktív szintézishez használható oszcillátor algoritmusok feltérképezése, összehasonlítása. Először ismertetem a szubtraktív szintézis elvét, az analóg szintetizátorok általános felépítését. Az oszcillátor megvalósításának triviális módszerén keresztül bemutatom a digitális implementáció során felmerülő legnagyobb nehézséget, az átlapolódást (aliasing). A szakirodalom alapján sorra veszem, és részletesen megvizsgálom a probléma megoldásának lehetséges módjait. Az ígéretesnek tűnő módszereket Matlab-ban implementáltam. A különböző oszcillátor algoritmusokat három csoportba sorolva mutatom be. Elsőként az ideális megoldással, az additív szintézissel foglalkozom, amely átlapolódástól mentes jelet állít elő. Kitérek a hullámtábla szintézisre, ami az additív szintézis egy változatának tekinthető. A második csoportba az ideális, sávkorlátozott megoldást közelítő módszerek tartoznak. Ebben a szakaszban a BLIT, BLEP és PolyBLEP megoldásokat vizsgálom. A harmadik csoportban szereplő algoritmusok a Lane, a DPW és a DPW2X, melyek az átlapolódást a spektrum esésének módosításával küszöbölik ki. Az algoritmusokat összehasonlítom, amihez néhány egyszerű kritériumot veszek figyelembe. Ezek a számításigény, a jel/zaj viszony, valamint a meghallgatásos teszt. Az összehasonlítás célja a megfelelő algoritmust kiválasztása, amelyet az implementációban alkalmazhatok. Rövidebben foglalkozom a szubtraktív szintézisben használt szűrő (Moog szűrő) digitális implementációjának kérdéseivel. A dolgozatot a legmegfelelőbbnek ítélt oszcillátor algoritmust és a vizsgált digitális szűrőt megvalósító VST alapú szoftveres szintetizátor bemutatása zárja.

Abstract

By now computers have become main platforms of creating and editing music, especially among amateur musicians for whom the expensive and room-demanding studio devices are not available. A personal computer, a mid-level quality sound card, and a few software give acceptable solution for this wide range of people. The demand for software applications aiding composing keeps growing. Among others, such applications are the virtual instruments, a lot of them being simulations of real instruments.

This thesis covers the questions of realizing a virtual analog synthesizer. The most common synthesis principle for analog synthesizers is subtractive synthesis which generates sound filtering a signal with rich spectral content. The central issue of this thesis is the review and comparison of oscillator algorithms suitable for subtractive synthesis. First, the principle of subtractive synthesis and the general structure of analog synthesizers are discussed. Through the trivial realization of an oscillator, the biggest difficulty of digital implementation, the aliasing, is pointed out. With the help of resources on this subject different ways of solving the problem are analyzed. The most promising methods are implemented in Matlab. The various algorithms are presented in three groups, starting with the ideal solution named additive synthesis which generates signals with no aliasing. Wavetable synthesis is outlined, a method that can be considered a variation of additive synthesis. The second group consists of methods approximating the ideal, band-limited solution. In this phase the BLIT, BLEP and PolyBLEP algorithms are examined. The third group contains the Lane, DPW and DPW2X methods. They avoid aliasing by modifying the spectrum tilt. Although the latter two groups do not completely eliminate aliasing, their computational cost is much less than that of additive synthesis. I compare the algorithms with the help of some simple criteria namely the computational cost, signal-to-noise ratio and the results of listening tests. In a short chapter the questions of digital implementation of the Moog low-pass filter are considered. Finally the implementation of a VST-based software synthesizer that applies the oscillator algorithm judged best and the digital Moog filter is described.

1. Bevezetés[i]

1.1 Történeti áttekintés

Szintetizátoron olyan elektronikus eszközt értünk, mely hangjelek mesterséges előállítására képes, analóg szintetizátoron pedig olyan szintetizátort, amiben a jel szintézise analóg módon, analóg áramkörök segítségével zajlik.

A 20. század elejétől kezdve egyre többen próbálkoztak hangszintézisre alkalmas eszközök megalkotásával, tömeges elterjedésükre azonban körülbelül az 1970-es évekig várni kellett. Ekkorra az elektronika fejlődése elérte azt a pontot, hogy olyan eszközök gyártására legyen lehetőség, amik már nem szekrény méretű monstrumok, akár hordozhatóak, áruk viszonylag megfizethető, és kezelésük sem igényel nagy műszaki felkészültséget.

A különböző kifejlesztett szintézis technikák közül az additív, valamint a szubtraktív szintézis voltak a legjellemzőbbek a korai analóg szintetizátorokban.

Az additív szintézis lényege, hogy az előállítandó jelet szinuszos oszcillátorok kimenetének összegeként állítjuk elő, vagyis az elérni kívánt frekvenciaképet diszkrét frekvenciakomponensek előállításával és összegzésével kapjuk meg.

A szubtraktív szintézis módszer egy erre a célra gondosan kiválasztott jel szűrésével hozza létre a kívánt hangot. Alapjelnek egy harmonikusokban gazdag jelet szokásos venni. Erre a célra a leggyakoribb jelek a négyszög-, háromszög-, és fűrészjel. A felsorolt hullámformával rendelkező periodikus jelek spektruma végtelen széles, így teljesül rájuk, hogy harmonikus tartalmuk nagy. Az alapjelből azután egy audiofrekvenciás szűrő segítségével vonjuk ki (innen a szubtraktív elnevezés) a számunkra nem kívánatos frekvenciakomponenseket, harmonikusokat.

A szubtraktív szintézis egyszerűsége és viszonylag könnyű megvalósíthatósága miatt a legnépszerűbb technika volt azokban az időkben, mikor az analóg szintetizátorok berobbantak a köztudatba. Ez az 1960-as évekre tehető, s nagyban köthető a Robert Moog amerikai mérnök által tervezett Moog szintetizátorok megjelenésének. Ebben az évtizedben kezdték el felfedezni a zenészek a szintetizátorokban rejlő lehetőségeket, alkalmazni őket szerzeményeik megalkotásában. 1971-ben megjelent a MiniMoog, amely elődjének könnyebben kezelhető, s ami fontosabb, kisebb, hordozható változata volt. Innentől kezdve a szintetizátorok térhódítása megállíthatatlan volt. Mivel a digitális szintetizátorok megjelenéséig tartó korszakban ez volt az uralkodó technika, analóg szintetizátor alatt általában egy a szubtraktív szintézis elvét alkalmazó eszközt értenek. Szakdolgozatomban én is ezt a technikát implementálom.

Később az integrált áramköri technika fejlődésével megjelentek a digitális szintetizátorok, melyek speciális célú, digitális jelprocesszort (DSP) alkalmazó számítógépeknek tekinthetőek. Előnyük, hogy stabilabb a működésük az analóg szintetizátorokkal szemben, melyeknek analóg áramkörei sokkal hajlamosabbak az „elhangolódásra”. Olcsóbbak, valamint újabb, korszerűbb szintézis módszerek (mint pl. FM szintézis, hullámtábla szintézis, hangminta alapú szintézis, fizikai modell alapú szintézis stb.) valósíthatóak meg rajta, melyek alkalmasabbá teszik valódi

[i] A Bevezetés megírásakor főleg [18], [14], [17] cikkekre támaszkodtam.

hangszerek hangjának szintetizálására. A digitális jelfeldolgozás hatékonysága a mikroprocesszorok számítási teljesítményének, gyorsaságának robbanásszerű növekedésének köszönhetően nagyságrendekkel jobb lett az analógénál. Ennek következménye a digitális szintetizátorok másik nagyon jelentős előnye is, hogy a polifónia sokkal könnyebben kivitelezhető velük, mint az analóg szintetizátorokkal. Az analóg szintetizátorok nagy korlátja volt, hogy szinte kizárólag monofónikusak voltak, egy-két kivételtől eltekintve. Ezeknél a kivételeknél a működést drasztikusan leegyszerűsítették (ez a termék minőségének, képességeinek csökkenésében mutatkozott), ily módon lehetővé téve, hogy több billentyűhöz is beépítsék ugyanazt az elektronikát a polifónia érdekében.

Az 1980-as évektől kezdve a digitális szintetizátorok uralják a piacot, háttérbe szorítva analóg elődeiket. Azok mégsem tűntek el teljesen, ugyanis egyrészt a klasszikus analóg hangzást a digitális szintetizátorok nem tudják tökéletesen visszaadni (az analóg áramkörök tökéletlenségeit, nemlinearitásait nem tudják teljességgel szimulálni), másrészt az analóg szintetizátorokra jellemző paraméterezhetőség, modularitás sokak számára még ma is vonzó.

Az 1990-es évektől kezdődően számos gyártó gyárt olyan digitális eszközöket, melyek kifejezetten a régi analóg szintetizátorok működését modellezik, szimulálják. Ezeket az eszközöket nevezik virtuális analóg szintetizátoroknak. A személyi számítógépek teljesítményének növekedésével pedig lehetővé vált, hogy a korábban célhardvert alkalmazó megoldások egy személyi számítógépen, szoftveresen is megvalósíthatóak legyenek. Ezeket nevezzük szoftveres szintetizátoroknak, közöttük sok olyat találhatunk, melyek szintén egy analóg szintetizátor viselkedését szimulálják. Ezeket is virtuális analóg szintetizátoroknak tekinthetjük. Napjainkban nagy népszerűségnek örvendenek a szoftveres szintetizátorok, ugyanis mára a számítógép a zeneszerzés, zeneszerkesztés állandó platformjává vált. A drága és nagy helyigényű, sok elemből álló stúdió berendezések helyett egy integrált, kompakt munkakörnyezetet állíthatunk össze mindössze egy PC, egy (az alaplapokra integráltaknál) jobb minőségű hangkártya, és a PC-n futó programok segítségével. Nagy a kereslet a szoftveres virtuális hangszerek, szintetizátorok iránt, ily módon a szoftveres virtuális analóg szintetizátorok iránt is.

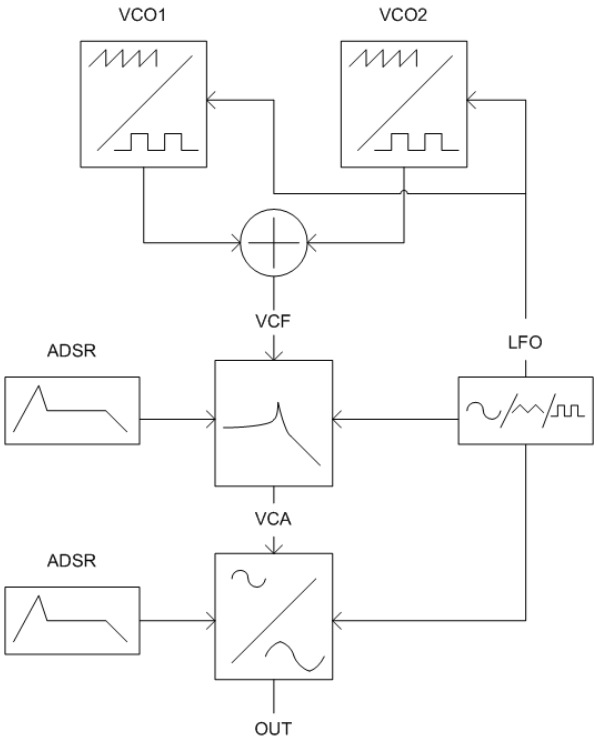
1.2 Felépítés

Egy analóg, szubtraktív szintézist alkalmazó szintetizátorban általában megtalálható egy vagy több oszcillátor, egy (sokszor rezonáns) szűrő, valamint valamilyen modulációs lehetőséget realizáló modul - leggyakrabban egy LFO, azaz kisfrekvenciás oszcillátor -, mellyel a többi modul paramétereit (pl. a szűrő vágási frekvenciája, a jelgenerátor amplitúdója, esetleg az oszcillátorok alapfrekvenciája) vezérelhetjük, modulálhatjuk. Nem ritkán beépítenek úgynevezett ADSR modulokat is, melyekkel pl. a szűrő vágási frekvenciájának, vagy a kimenet hangerejének adhatunk burkológörbét, s szabályozhatjuk azt.

Ami az analóg szintetizátort vonzóvá teszi a zenészek számára, az a tény, hogy a modulok paraméterei az eszközön állíthatóak, tehát a felhasználó élesben „keverheti ki” az általa használni kívánt hangszínt. Nem ritkák a teljesen moduláris felépítésű eszközök, ahol a fenti modulokból több van, s szabadon egymás után „kötögethetőek”, variálhatóak, sok lehetőséget kínálva ezzel a zenésznek.

Az 1.1-es ábrán egy tipikus szubtraktív szintézis elvű analóg szintetizátor blokkdiagramja látható. Az ábrán szereplő rövidítések jelentése:

- VCO : Voltage Controlled Oscillator feszültségvezérelt oszcillátor
- VCF : Voltage Controlled Filter feszültségvezérelt szűrő
- VCA : Voltage Controlled Amplifier feszültségvezérelt erősítő
- LFO : Low Frequency Oscillator alacsony frekvenciás oszcillátor
- ADSR: Attack time, Decay time, Sustain level, Release time
  burkológörbe paraméterek (felfutási- , leereszkedési- (az alap jelszintre) idő, alap jelszint, felengedési idő)



.. ábra Tipikus, szubtraktív szintézist alkalmazó analóg szintetizátor felépítése

1.3 A célok megfogalmazása

Szakdolgozatom célja egy virtuális analóg szintetizátor szoftveres implementációja. A szintetizátornak a következő modulokat, funkciókat kell tartalmaznia:

- két oszcillátor, melyek mindegyikénél választhatunk két hullámforma (négyszög, fűrész) közül.
- egy rezonáns aluláteresztő szűrő
- egy kisfrekvenciás oszcillátor (Low Frequency Oscillator - LFO), mellyel modulálhatjuk a kimenet amplitúdóját (tremoló), az oszcillátorok frekvenciáját (vibrato), valamint a szűrő vágási frekvenciáját
- képesnek kell lennie a polifónikus üzemre

Az itt leírt funkcionalitás körülbelül megegyezik a klasszikus MiniMoog analóg szintetizátor funkcionalitásával, azzal a két fő különbséggel, hogy az monofónikus volt, valamint tartalmazott ADSR funkciót.

Kiemelt figyelmet fordítottam az oszcillátorok digitális megvalósításával kapcsolatos kérdésekre, a dolgozat legnagyobb részét az aliasing jelenség által okozott probléma megoldásának vizsgálata teszi ki. A különböző algoritmusokat, melyek a fenti problémára kínálnak megoldást, Matlab-ban implementáltam, teszteltem. Részletes vizsgálatuk, elemzésük célja, hogy kiválasszam a célra legmegfelelőbbet, melyet a program megvalósítása során alkalmazhatok. A választást egy egyszerű kritériumrendszer felállításával segítettem. A kritériumok alapján a cél egy olyan algoritmus kiválasztása, melynek számításigénye kicsi, kielégítő hangminőséget nyújt, és egyszerűen megvalósítható vele a frekvencia modulációja az LFO beépítésével. A minél jobb hangminőség és a kis számítási igény egymásnak ellentmondó követelmények lehetnek, itt megfelelő kompromisszumra törekedtem.

A szintetizátort a Steinberg cég által kifejlesztett Virtual Studio Technology (VST) segítségével implementáltam. A VST egy ingyenesen használható keretrendszer, melynek segítségével úgynevezett plugin-okat hozhatunk létre (olyan alprogramokat, melyek egy másik program alatt képesek futni). Ezeket azután egy VST plugin-ok futtatására képes programban futtathatjuk, használhatjuk.

A szintetizátor MIDI billentyűzet segítségével vezérelhető. A VST környezet lehetőséget biztosít a MIDI információk kezelésére.[ii]

[ii] MIDI: Musical Instrument Digital Interface – aszinkron soros átviteli protokoll, melyet elektronikus zenei eszközök közötti kommunikációra használnak

2. Oszcillátor algoritmusok bemutatása, elemzése, összehasonlítása

2.1 Bevezetés

Ebben a fejezetben a virtuális analóg szintetizátor központi elemének, az oszcillátornak a digitális megvalósítási lehetőségeit elemzem. Ismertetem a felmerülő legnagyobb megoldandó problémát, az aliasing jelenséget, s annak kiküszöbölési lehetőségeit. Bemutatom a megoldás különböző megközelítési módjait, s az ezekhez kapcsolódó algoritmusokat. A fejezet végén egy egyszerű kritériumrendszer alapján összehasonlítom az algoritmusokat, s kiválasztom azt, melyet a kritériumok alapján a legmegfelelőbbnek ítélek, s implementálok a programban.

Az alapprobléma az aliasing nevű jelenség. A szubtraktív szintézishez - melyet a szintetizátorban alkalmazok - egy harmonikusokban gazdag jel előállítása szükséges. Az erre a célra leggyakrabban alkalmazott jelek a négyszög-, háromszög-, valamint fűrészjelek. Ezek mindegyike végtelen széles spektrumú, a végtelen frekvenciákon is rendelkezik frekvenciakomponenssel (elméletben – gyakorlatban megvalósított rendszerekben ez nyilván lehetetlen). A négyszög- és fűrészjel harmonikusainak amplitúdói a frekvencia növekedésével $1/f$, a háromszögjelé $1/f^2$szerint csökkennek. Szoftveres szintetizátorról lévén szó, ez egy diszkrét idejű, tehát mintavételes rendszer. Nyquist és Shannon mintavételi tételéből tudjuk, hogy egy folytonos idejű jel mintavételezéssel kapott diszkrét idejű reprezentációja csak akkor alakítható vissza hiba nélkül eredeti állapotába, ha az eredeti folytonos idejű jel nem rendelkezik a mintavételi frekvencia felénél nagyobb frekvenciakomponenssel. Ellenkező esetben a mintavételi frekvencia felénél nagyobb komponensek átlapolódnak, ezt nevezzük aliasing-nak.

Ha tehát oszcillátoraink jeleit egyszerűen folytonos idejű megfelelőik mintavételezett verzióiként állítjuk elő (ezt naiv, vagy triviális megoldásnak nevezzük), az átlapolódott komponensek miatt minőségromlást fogunk tapasztalni. Hangkártyánk D/A átalakítója, annak aluláteresztő szűrője ugyanis csak a mintavételi frekvencia (a továbbiakban: f_s) fele feletti frekvenciákat fogja eltávolítani, az $f_s/2$ alá átlapolódott komponenseket nem. Olyan megoldást, algoritmust kell tehát keresnünk, amely sávkorlátozott jelet állít elő, vagy legalábbis mérsékli az aliasing által okozott hibát annyira, hogy az emberi fül számára az ne legyen észrevehető.

A különböző megoldásokat, algoritmusokat a fűrészjel előállításán keresztül fogom bemutatni, majd a későbbiekben ismertetem, hogyan lehet más hullámformákat abból (vagy közvetlenül) előállítani.

A lehetséges módszereket [1] alapján három csoportba sorolhatjuk:

I. **Tökéletesen sávkorlátozott megoldások**: ide azok a módszerek tartoznak, amelyek olyan jelet állítanak elő, amely egyáltalán nem tartalmaz frekvenciakomponenst $f_s/2$ fölött.
II. **Kvázi-sávkorlátozott megoldások**: ezek a módszerek aluláteresztő szűrővel szűrt jelet állítanak elő. Az átlapolódás mértékét a szűrés pontossága, minősége határozza meg.
III. **A spektrum „esését” módosító megoldások**: ezek a módszerek olyan jelet állítanak elő, aminek a spektruma meredekebben csökkenő a mintavételezés előtt. A mintavételezés után egy szűrővel korrigálják a spektrumot a megfelelő amplitúdómenet helyreállítására.

Az I. csoportba tartoznak az additív szintézist, valamint annak változatait (pl. hullámtábla szintézist) alkalmazó megoldások.

A II. csoportba a Band-limited Impulse Train (sávkorlátozott impulzus-sorozat), röviden BLIT nevű módszer és az ugyanarra az elvre épülő, de számításigény szempontjából hatékonyabb Band-limited Step Function (BLEP - sávkorlátozott lépésfüggvény) és változatai (MinBlep, PolyBlep) tartoznak.

A III. csoportba sorolható a Lane-módszer, és a DPW algoritmus (Differentiated Parabolic Wave – differenciált parabolikus hullám) valamint annak változatai.

A fentieken kívül egyéb módszerek is léteznek/létezhetnek, azokat soroltam fel, amelyekről szakirodalmat találtam, melyeket megvizsgáltam, és amelyekről a szakdolgozatban részletesen szó esik. Az ígéretesnek tűnő algoritmusokat (Additív, Lane, DPW, DPW2X, PolyBLEP), valamint viszonyítási alapként a triviális megoldást Matlab-ban implementáltam.

A módszerek összehasonlítását négy fő kritérium alapján végzem:

- számításigény, az algoritmus bonyolultsága, rendszerint annak megbecslése, hogy egy minta kiszámítása mennyi (és milyen) műveletet vesz igénybe
- jel/zaj viszony - az erre alkalmazott számítási módszert később bemutatom
- meghallgatásos teszt – egyszerűen meghallgatom, hogyan szólnak a különböző algoritmusokkal előállított jelek (itt nyilván egy kellően nagyszámú alany bevonásával végzett teszt hozna statisztikailag szignifikáns eredményt, sajnos erre nem volt lehetőség).
- az algoritmus robosztussága a modulációkra nézve – egy önmagában egyszerű algoritmus implementálása komplexszé, bonyolulttá válhat, ha pl. a frekvencia, vagy az amplitúdómodulálási lehetőségét (pl. LFO-val) is bele szeretnénk építeni a szintetizátorba

A későbbi fejezetekben elemzett megoldásokat a MathWorks cég Matlab (R2006-os verzió) programjában implementáltam, s annak segítségével vizsgáltam (a spektrumokat és jelformákat ábrázoló ábrák is ezzel a programmal készültek).

2.2 A triviális megoldás és az aliasing

A triviális fűrészjel

Kézenfekvő megoldásnak tűnik a fűrészjelet az analóg jel mintavételezett verziójaként előállítani (ezt szokás naiv, avagy triviális megoldásnak nevezni a szakirodalomban). Ekkor a fűrészjelet előállíthatjuk egy egyszerű számláló kimeneteként. A kimenet mintáit ez esetben a következő képlet adja [2]:

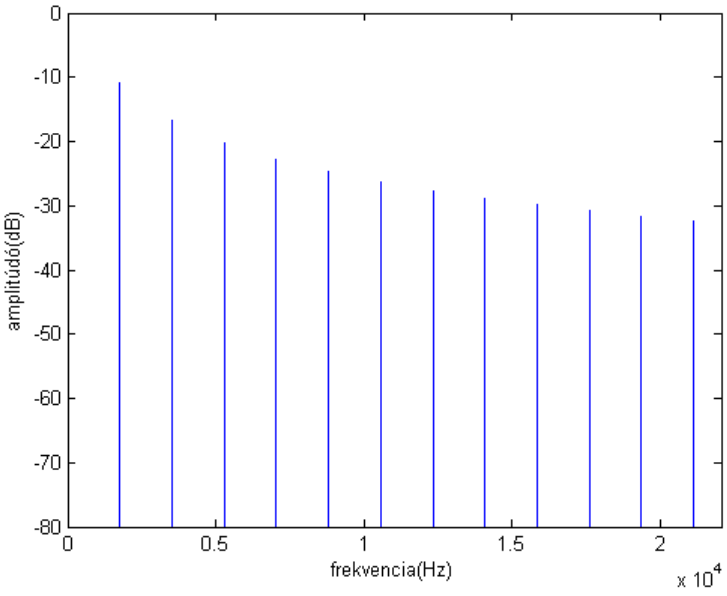
$$s[n] = 2\left(n\frac{f}{f_s}\ mod\ 1\right) - 1 \qquad (2.1)$$

ahol s a kimeneti minta értéke, n a diszkrét idő jele, f_s a mintavételi frekvencia és f a fűrészjel periódusideje, az alapharmonikus frekvenciája. A zárójelben szereplő $\frac{f}{f_s}$ az az érték, amennyivel a számláló értéke két minta között növekszik, a mod pedig maradékképzést jelent. A zárójel előtti 2-es szorzó valamint az ez utáni kivonás -1 és 1 közé skálázza a jel értékeit. Erre azért van szükség, mert a hangkártyák ebben az értéktartományban fogadják a mintákat. Az e tartományon kívül eső értékeket a hangkártya úgy értelmezi, mintha a tartomány széleinek értékeit vennék fel. Szemléletesen: „levágja" a tartományon kívül eső részt (clipping). Ez a jelben torzítást eredményez, ami kerülendő.

A triviális megoldás az úgynevezett aliasing hibához vezet, melyet a következő ábrák illusztrálnak. A 2.1-es, 2.2-es, 2.3-as ábrákon látható sorrendben az ideális (folytonos idejű) fűrészjel spektruma, a triviális módszerrel előállított fűrészjel spektruma, valamint a triviális és az ideális fűrészjel különbségének spektruma, tehát csak a hiba.

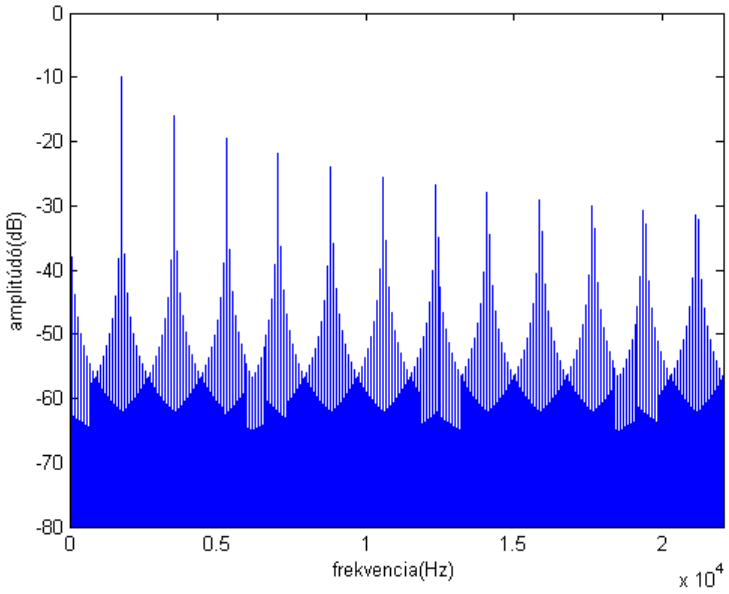
Az alapfrekvencia a harmadik vonalas „A" zenei hang frekvenciája, 1760 Hz. A mintavételi frekvencia 44100 Hz.[iii]

Láthatjuk, hogy a triviális módszer korántsem az ideálissal egyező frekvenciatartalmú jelet ad. Ez nem kívánt hatásokkal jár. Azon túlmenően, hogy nem a kívánt jelet kapjuk, a fül számára zavaró, zaj jellegű hanghatásokat észlelünk a kívánt harmonikusok mellett.

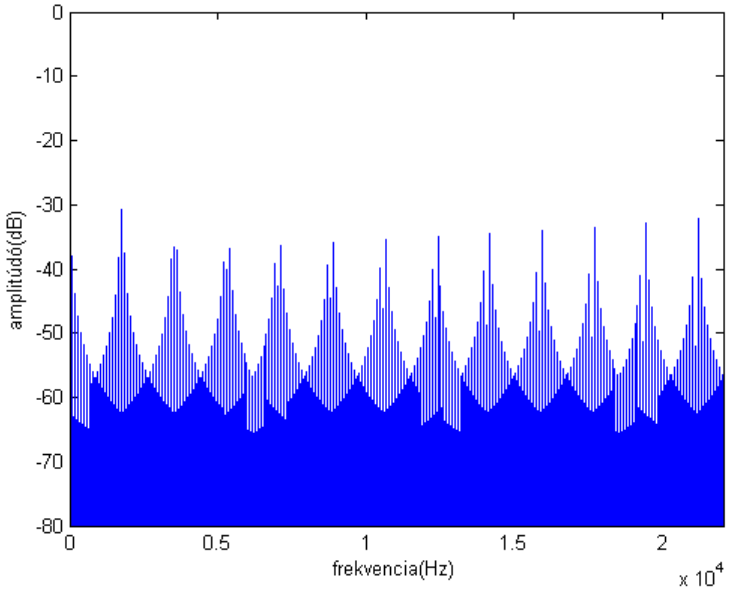


.. ábra Az ideális fűrészjel spektruma

[iii] Ez a mintavételi frekvencia általánosnak mondható a hangtechnikában (többek közt a CD technikában is ezt alkalmazzák [4]).



.. ábra A triviális megoldással előállított fűrészjel spektruma



.. ábra Az ideális és a triviális fűrészjel különbségének spektruma (az aliasing miatt létrejövő frekvencia komponensek)

Az aliasing megjelenésének okáról, [3] alapján

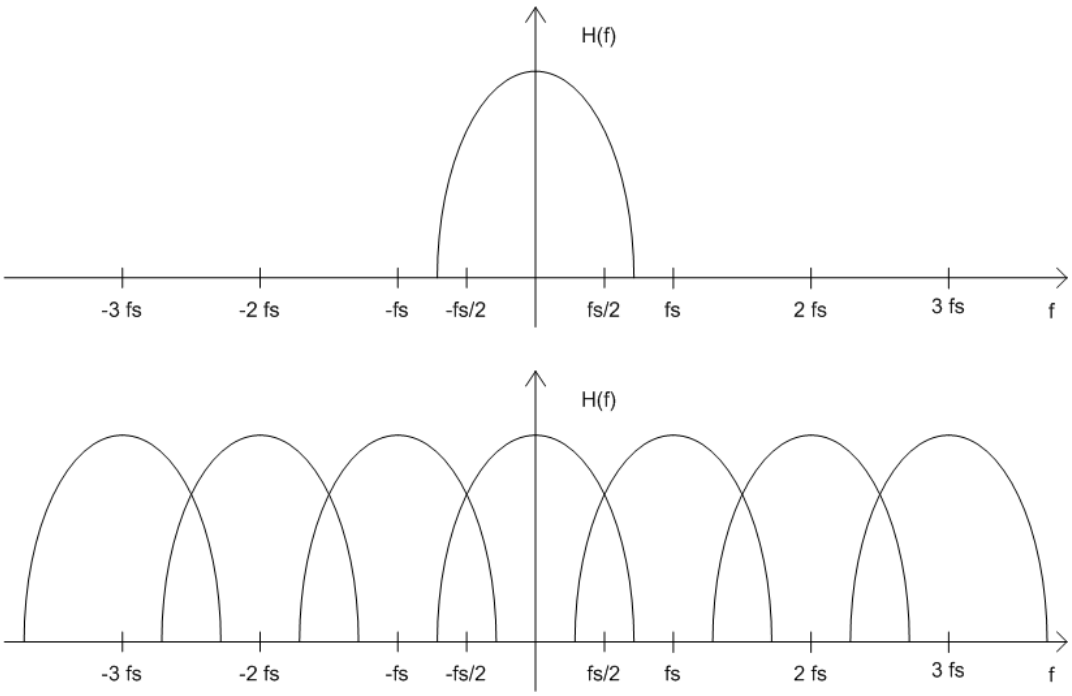
A matematikai értelemben vett mintavételezés azt jelenti, hogy megszorozzuk a mintavételezendő folytonos idejű jelet egy olyan Dirac-impulzussorozattal, ahol az impulzusok időbeli távolsága a mintavételi periódus. Ha két jelet az időtartományban összeszorzunk, az ekvivalens azzal, hogy a frekvenciatartományban konvolváljuk őket. Az időtartománybeli Dirac-impulzussorozat a frekvenciatartományban is Dirac-impulzussorozat képét mutatja, az impulzusok távolsága f_s . Ha egy jelet egy ilyen impulzussorozattal konvolválunk, az szemléletesen úgy írható le, hogy a jelet mindegyik impulzusra „rámásoljuk”. Így a

mintavételezés során a mintavételezett jel spektruma az eredeti spektrumából úgy képezhető, hogy (a pozitív és negatív frekvenciák irányában egyaránt) a mintavételi frekvencia egész számú többszöröseire eltoljuk azt, és összegezzük az összes ilyen „eltolás” eredményét. Ebből következik, hogy ha a mintavételezett jel rendelkezett frekvenciakomponenssel $f_s/2$ felett, akkor átlapolódást tapasztalunk. Az átlapolódott komponensek az eredeti komponensek „másai” (alias – innen az elnevezés). A mintavételi frekvencia fele alá belapolódott komponensek nem kívánt komponensek, hibát jelentenek, hiszen meghamisítják az eredeti jelet. Az ilyen jelet analóggá visszakonvertálva (mely során az $f_s/2$ feletti frekvenciákat aluláteresztő szűrővel kiszűrjük) nem az eredetit fogjuk visszakapni.

A 2.2-es képlet, valamint a 2.4-es ábra szemlélteti a mintavételezés hatását a spektrumra.

$$X_s(f) = \sum_{k=-\infty}^{\infty} X(f - k \cdot f) \qquad (2.2)$$

A képletben $X(f)$ a folytonos idejű jel spektrumát, míg $X_s(f)$ a mintavételezett jelét jelöli.



.. ábra A mintavételezés hatása a spektrumra: az aliasing megfigyelhető (felül a mintavételezés előtti spektrum, alul a mintavételezés utáni; az ívek a jel sávkorlátját illusztrálják, a ténylegesen egymásra lapolódó komponenseket természetesen összegezni kell)

Az aliasing hatása a hangminőségre

A legnyilvánvalóbb hatás, hogy azok a belapolódott komponensek, amelyek intenzitása a hallásküszöb fölött van, hallhatóak lesznek, s jelentősen torzítják az eredeti hangképet. Ráadásul – egy kivételes esettől eltekintve – inharmonicitást jelentenek, amely a fül számára rendkívül

zavaró lehet (különösen kivehetőek, hallhatóak azok a komponensek, melyek az alapharmonikus alatt helyezkednek el).

A kivételes eset az, mikor a jel alapharmonikusának frekvenciája osztója a mintavételi frekvenciának. Ez a hangkép torzulásának egy másik esete. Ekkor az átlapolódott komponensek pontosan az eredeti komponensekre esnek. Ez esetben nem kapunk új komponenseket, inharmonicitást, viszont az eredeti amplitúdóértékek megváltoznak. Tehát amplitúdóhibát kapunk, ami szintén a hangszín változásával jár, habár a fül számára ez jóval kevésbé észrevehető, zavaró.

Előfordulhat, hogy egy átlapolódott komponens és egy harmonikus frekvenciatávolsága nagyon kicsi, és az amplitúdóik is hasonló nagyságúak. Ekkor a lebegés nevű jelenséget tapasztalhatjuk.[iv]

Abban az esetben, ha az alapfrekvencia változik (pl. LFO modulálja – vibrato), némely átlapolódott komponens az ellenkező irányba „mozdul el" a frekvenciatengelyen az alapharmonikushoz képest [1]. Ez szintén nemkívánatos dolog. Az angol nyelvű szakirodalomban heterodyning-nak nevezik a jelenséget (utalva a rádiótechnikában használatos heterodin vevőkészülékek működési elvére).

A triviális megoldás számítási igénye a lehető legkisebb, egy-egy minta kiszámításához mindössze egy számlálót kell növelni, ami egy darab összeadási művelet. Hiába azonban a számítási hatékonyság, ha – a fentebb említett jelenségek által okozott hangminőségbeli romlásnak köszönhetően – használhatatlan ez a módszer. Csak azokban az esetekben tekinthető viszonylag elfogadhatónak, ahol az alapfrekvencia nagyon kicsi a mintavételi frekvenciához képest, hiszen az minél kisebb, annál magasabb rendű harmonikusai (helyesebben azok alias-ai) lapolódnak be $f_s/2$ alá, így annál kisebb lesz az átlapolódott komponensek energiája.

Ebből is kikövetkeztethető, hogy az aliasing hiba kiküszöbölésének, csökkentésének legkézenfekvőbb módja a mintavételi frekvencia növelése, a túlmintavételezés. Ahhoz azonban, hogy használható eredményt kapjunk, az eredeti mintavételi frekvencia sokszorosát kellene használnunk, ami jelentős mértékben lecsökkentené az algoritmus hatékonyságát. Olyan algoritmusoknál azonban, ahol a hangminőség már az eredeti *fs* esetén is nagyságrendekkel jobb a triviális megoldásénál, lehet a túlmintavételezés használatának létjogosultsága, a minőség további javítása érdekében. Ekkor akár egy 2, vagy 4-szeres túlmintavételezés is elég lehet a kívánt eredmény eléréséhez. A későbbiekben mutatok példát a túlmintavételezés használatára.

[iv] Lebegés: két közeli frekvenciájú hang együttes megszólaltatásakor egy periodikusan ingadozó erősségű hangot hallunk. Ezt a jelenséget lebegésnek nevezzük. A lebegés frekvenciája megegyezik a megszólaltatott hangok frekvenciáinak különbségével. [15]

2.3 Tökéletesen sávkorlátozott megoldások

Additív szintézis

Az ideálisnak tekinthető megoldás az aliasing problémára az additív szintézis módszere. Ennek során egyesével hozzuk létre a kívánt harmonikusokat, majd összeadjuk őket. Abban az esetben, ha a létrehozni kívánt jelnek csak $f_s/2$ alatti komponenseit számítjuk ki, a megoldás ideális eredményt ad.

Mint minden periodikus jel, a fűrészjel is Fourier-sorba fejthető. Az additív szintézishez a jel Fourier-sorának képletét alkalmazzuk:

$$s[n] = -\frac{2}{\pi}\sum_{k=1}^{\infty}\frac{\sin\left(2\pi kn\frac{f}{f_s}\right)}{k} \tag{2.3}$$

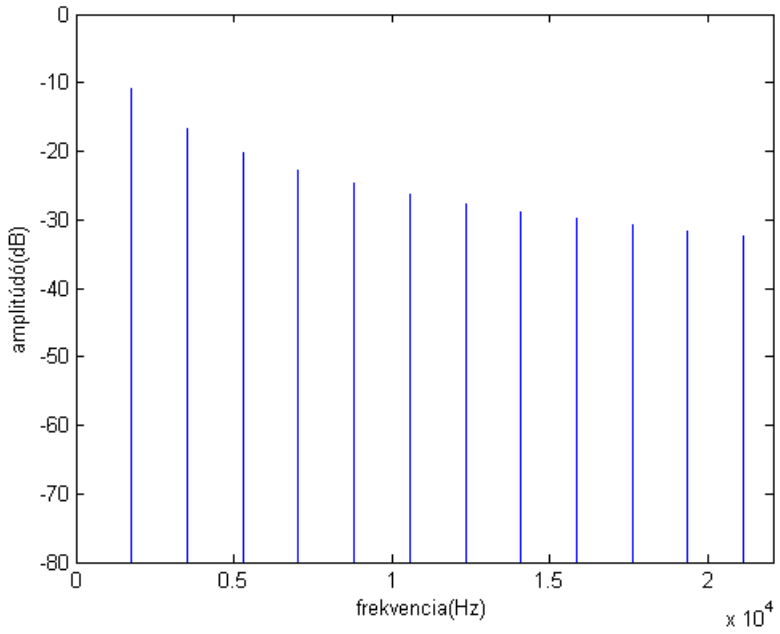
ahol s a kimeneti minta értéke, n a diszkrét idő jele, f_s a mintavételi frekvencia és f a fűrészjel periódusideje, az alapharmonikus frekvenciája. Ha *k*-t 1-től csak $\frac{f_s}{2f}$ egész részéig „futtatjuk", a legnagyobb hozzáadott harmonikus is a Nyquist-frekvencia alatt marad, nem jön létre aliasing.[v]

Ez a módszer, mint látjuk, hangminőség szempontjából ideális, hiszen teljességgel kiküszöböli a problémánkat, a gyakorlatban mégsem igazán használható. Az additív szintézis bizonyos szempontból a triviális megoldás ellenpárja, ugyanis amíg az minimális számításigényű, de használhatatlan hangminőséget produkál, addig az additív szintézis maximális minőséget nyújt, cserébe a számításigény (az egészen magas alapfrekvenciájú hangokat kivéve) viszont rendkívül nagy. Egy minta kiszámításához annyi szinuszos tagot kell összegeznünk, ahány harmonikusa az alapfrekvenciának $f_s/2$ alatt van. Ez K darab ilyen harmonikus esetén K darab szinusz érték számítását és K-1 darab összeadást jelent. Egy-egy oszcillátor érték, vagyis szinusz számítása ráadásul nagyobb műveletigényű, mint egy számláló iterálása (ami egy összeadásnak felel meg). Az is megfigyelhető, hogy a számításigény nem konstans (vagyis nem kalkulálható előre), hiszen függ az alapfrekvenciától (a szintetizátoron lejátszott hangtól): kisebb f több harmonikus - több számítás, és fordítva.

A triviális illetve az additív módszer a számítási komplexitás és a hangminőség szélsőértékeit képviselik tehát: egyik minimális komplexitású, a másik maximális minőségű (vagy ha úgy tetszik minimális komplexitás – minimális aliasing hiba/maximális számítási hatékonyság – maximális minőség). Olyan algoritmust kell találnunk, mely elfogadható kompromisszumot jelent e két egymással szemben álló követelmény között.

A következő ábrán az additív szintézissel előállított fűrészjel spektrumát ábrázoltam. Látható, hogy nincsen aliasing, csak a harmonikusok jelennek meg.

[v] A 2.3-as képletben található mínusz szorzó arra szolgál, hogy emelkedő fűrészjelet kapjunk, elhagyásával nyilván csökkenő fűrészjel jön létre.



.. ábra Az additív szintézissel előállított (ideálisan sávkorlátozott) fűrészjel spektruma (alapfrekvencia f=1760 Hz)

Hullámtábla szintézis

A hullámtábla (wavetable) szintézis az additív szintézis egy változata. Ez a módszer az additív szintézis elvét alkalmazza a jel előállítására, de a számításigényt csökkenteni próbálja olyan módon, hogy a mintaértékeket nem valós időben, hanem előre kiszámoljuk, majd a memóriában (vagy valamilyen adathordozón) eltároljuk. A tárolt adathalmaz a hullámtábla. Ekkor a jel előállítása a hullámtábla periodikus kiolvasásából áll.

Egy hullámtábla többnyire a jel egy periódusát tartalmazza. A tábla értékeit additív szintézissel az összes kívánt harmonikust összegezve képezzük. A lejátszott hang alapfrekvenciája f_s/N, ahol N a hullámtábla mérete mintákban, ha a kiolvasás sebessége f_s. Ha pl. az emberi hallásküszöb alsó határának tekintett 20 Hz-es hangot szeretnénk szintetizálni, és a mintavételi frekvencia 44.1 kHz, akkor N=2205. Ha egy-egy mintát 16 biten – azaz 2 byte-on – ábrázolunk (ahogy pl. a CD szabványban is teszik [4]), akkor ez egy $2205 \cdot 2 = 4410$ byte méretű táblát eredményez. Ez ugyan nem sok, de ez csak egyetlen hang, ha ezt sok hangra külön-külön megvalósítjuk, ez a méret a sokszorosára emelkedhet. Ez bizonyos architektúrák esetén, ahol a felhasználható memória erősen korlátolt, problémát jelenthet, PC-n természetesen nem. Észre kell venni azonban, hogy csak olyan hangokhoz tudunk hullámtáblát létrehozni ily módon, melyeknél N egész értékűre adódik, hiszen tört méretű táblát nem tudunk megvalósítani.

Felmerül a gondolat, hogy spórolni tudunk a táblák számával, ha figyelembe vesszük, hogy ha pl. minden második mintát játszunk csak le, akkor kétszeres alapfrekvenciájú hangot kapunk, ha minden harmadikat, akkor háromszorost, és így tovább. Számolnunk kell azonban azzal, hogy ekkor ismét megjelenik az aliasing jelenség, hiszen ha pl. minden második mintát játsszuk le, az annak felel meg, mintha a mintavételi frekvencia az addigi fele lenne. Ebben az esetben csak akkor kerüljük el az átlapolódást, ha a táblába csak az eredeti f_s negyedéig generálunk

harmonikusokat. Ha ennek ellenére mégis ezt a megoldást választjuk (pl. mert úgy döntünk, a legmagasabb harmonikusokra nincs szükségünk és/vagy kismértékű átlapolódást megengedünk), akkor is gond, hogy csak korlátolt mennyiségű hangot tudunk létrehozni, f_s/N egész számú többszöröseinek megfelelő alapfrekvenciájúakat. Erre megoldás, ha megengedünk nem egész „lépésközöket" is. Ekkor, ha a kijátszandó minta indexe tört (tehát a hullámtábla két eleme közé mutat), azt kezelnünk kell valamilyen módon. Egy lehetőség, hogy a tört indexeket egésszé kerekítjük, tehát mindig a „legközelebbi" táblaelemet olvassuk ki (nagyon nagy N, vagy sokszorosan túlmintavételezett jel esetén van értelme [3]). Egy másik lehetőség pedig, hogy interpolálunk. Ekkor az interpoláció minősége fogja meghatározni, hogy mennyire közelítjük meg a tökéletes sávkorlátozottságot (természetesen azon kívül, hogy mekkora frekvenciáig generáljuk a harmonikusokat). Sokfajta módszert alkalmazhatunk, a lineáris interpolációtól kezdve a teljesen sávkorlátozott interpolációig (sinc interpoláció). E kettő jelenti a két szélsőértéket bonyolultságban, és minőségben egyaránt. Ha ezt a módszert alkalmazzuk, itt is kompromisszumra kényszerülünk tehát.

Az interpoláció alkalmazása jelentősen bonyolítja az algoritmust, és megnöveli a számításigényt minden egyes minta számításánál, ahol interpolációra szükség van (márpedig az ilyen minták vannak nagy többségben). Ha kielégítő minőségű interpolációt szeretnénk, az algoritmus olyan bonyolult lesz (az eredeti koncepció – miszerint ne valós időben számoljuk a mintákat, hanem tároljuk előre – szinte teljesen elvész), hogy nagyságrendileg nem lesz jobb a szimpla additív szintézisnél, ami viszont nagy számításigénye ellenére legalább ideális minőséget ad. A hullámtábla módszerrel ezért részletesebben tovább nem foglalkozom. [vi]

2.4 Kvázi-sávkorlátozott megoldások: a BLIT-től a PolyBlep-ig

Majdnem teljesen átlapolódástól mentes jelet állíthatunk elő az ebben az alfejezetben taglalt módszerekkel. Az itt elemzett megoldások koncepciója, hogy a folytonos idejű jelet először egy aluláteresztő szűrésnek vessük alá, eltávolítva (de legalábbis jelentősen csillapítva) ezáltal a Nyquist-frekvencia feletti komponenseket. Ekkor a hangminőség szűk keresztmetszete a szűrés minősége lesz.

Az elv az, hogy képzeletben a mintavételezés pillanata előtt szűrjük a jelet. Olyan algoritmust használunk, ami már egy aluláteresztővel szűrt folytonos idejű jelnek állítja elő a diszkrét idejű reprezentációját. Ilyen módszer a BLIT, és a belőle származtatottak, mint a BLEP és a PolyBlep.

A BLIT módszer

A BLIT rövidítés a Bandlimited Impulse Train (sávkorlátozott impulzussorozat) kifejezést takarja, amiből kiderül, hogy egy impulzussorozat sávkorlátozott megvalósítására ad megoldást. Ugyan az impulzussorozat nem az a hullámforma, amit szintetizátorokban közvetlenül alkalmazni lehetne, de generálhatóak belőle azok a hullámformák, amelyeket alkalmazni szeretnénk (fűrészjel, négyszögjel, háromszögjel) [3].

[vi] Sok helyen hullámtábla szintézis néven említik a PCM (Pulzus kód moduláció) szintézist is, azonban az némileg más elven alapul: ott a „hullámtáblát" digitális hangrögzítéssel felvett hangminták (PCM sample) alkotják.

$$IT(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT) \tag{2.4}$$

A 2.4-es képlet egy folytonos idejű, egységnyi amplitúdójú impulzussorozatot ír le, *t* a folytonos idő jele, *T* pedig a két impulzus közötti időbeli távolság. $\delta(\tau)$ a folytonos idejű Dirac-impulzus.

Fűrészjelet az impulzussorozatból annak integrálásával kaphatunk:

$$s_{fűrész}(t) = -\left(1 + 2\int_0^t IT(\tau) - C d\tau\right) \tag{2.5}$$

A képletben IT az impulzussorozatot jelöli, melyből egy konstanst ($C = \int_0^T \frac{IT(\tau)}{T} d\tau$, az impulzussorozat egy periódusra vett átlaga) ki kell vonni, hogy az integrálás ne a végtelenbe tartson, hanem egy fűrészjelet adjon. Ez a konstans – ahogy a képletből is látszik – frekvenciafüggő. *T* jelölte az impulzusok időbeli távolságát az impulzussorozatban, az integrálás után *T* a fűrészjel periódusidejének felel meg. Az integrál kifejezés egy 0 és -1 közötti csökkenő fűrészjelet hoz létre. Az integrál szorzása kettővel, az 1 hozzáadása, majd az egész negálása skálázza a jelet -1 és 1 közötti emelkedő fűrésszé. A képletben IT, az impulzussorozat helyére annak sávkorlátozott verzióját (BLIT) helyettesítve sávkorlátozott fűrészjelet kapunk.

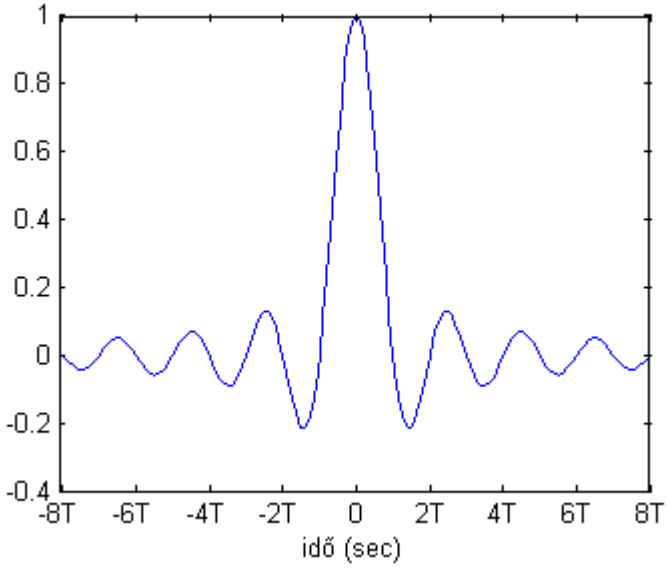
Olyan diszkrét idejű jelet kell létrehoznunk, mely aluláteresztő szűrésen átesett folytonos idejű jel mintavételes reprezentációja. A 2.4-es képletben szereplő impulzussorozatot szeretnénk szűrni. A szűrés a szűrő impulzusválaszával való konvolúciót jelent az időtartományban. A konvolúció – minthogy a jel $t=kT$-t kivéve 0 – itt gyakorlatilag azt jelenti, hogy az egyes impulzusokat helyettesítjük a szűrő impulzusválaszával, s összegzünk. Rajtunk múlik, milyen szűrőtervezési technikát alkalmazunk. A létrehozott szűrő (annak impulzusválasza, melyekkel az impulzusokat „helyettesítjük" a konvolúció során) fogja meghatározni az algoritmusunk bonyolultságát, és minőségét egyaránt.

Az ideális aluláteresztő szűrő impulzusválasza egy sinc függvény ($sinc(x) = \frac{\sin(\pi x)}{\pi x}$). Egy $f_s/2$ vágási frekvenciájú ideális aluláteresztő szűrő impulzusválasza:

$$h(t) = f_s \frac{\sin(\pi f_s t)}{\pi f_s t} \tag{2.6}$$

Kézenfekvő megoldás, hogy egy sinc függvényt (2.6-os ábra) másoljunk egy-egy impulzus helyére, s ezt a jelet mintavételezzük. Ekkor tulajdonképpen az ideális aluláteresztő szűrő impulzusválaszának mintavételezett változatát vesszük (ami a folytonos idejű rendszer impulzus-invariáns transzformációjának felel meg a diszkrét tartományba). Több problémával is szembetaláljuk magunkat azonban.

Az egyik, hogy a sinc függvény (az ideális aluláteresztő impulzusválasza) időben végtelen pozitív és negatív irányba egyaránt. Ez azt jelenti, a gyakorlatban csak egy bizonyos szakaszát tudjuk használni. Ekkor viszont a szűrés már nem lesz ideális. Ablakoznunk kell tehát az impulzusválaszt, ezt a technikát hívják BLIT-SWS-nek (Sum of Windowed Sincs). El kell döntenünk, hogy hány minta szélességű és milyen paraméterekkel rendelkező ablakot választunk. Minél több minta szélességű ablakot választunk, nyilván annál nagyobb lesz az algoritmusunk számításigénye.



.. ábra Ideális (sinc) aluláteresztő szűrő impulzusválasza (nulla-átmenetek T időközönként; T=1/fs, a szűrő vágási frekvenciája fs/2)

Egy másik probléma, hogy ha az előállítandó jel periódusa nem egész számú mintából áll. Ez mindig teljesül, ha az alapfrekvencia nem osztója f_s-nek. Ekkor az impulzusok „két minta közé” esnek. Ilyenkor a szűrő impulzusválaszának különböző fázisú mintavételezéseit kell megvalósítani, ami alatt a mintavételi pontok eltolását értem. Attól függően, hogy az adott impulzus helye a két mintavételi pont között hol lenne, az impulzusválasz különböző mértékben eltolt változatát „másoljuk” az impulzus helyére. Szemléletesen ez úgy tekinthető, mintha a mintavételi pontok egy léckerítés azon pontjai lennének, ahol „belátunk a kertbe”, vagyis az impulzusválaszra. Az, hogy a mintavételi pontokat mennyivel toljuk el, attól függ, mekkora a mintákban vett periódusidő törtrésze (a két mintavételi pont között hová esik az impulzus).

Ez a megoldás úgy valósítható meg, ha például az eredeti impulzusválasz mellett kiszámoljuk annak negyed, fél, háromnegyed mintával eltolt (a mintavétel pontját toljuk el) verzióit is. Ekkor van 4 úgynevezett águnk. Ezután az alapján, hogy az impulzus a két minta között hol helyezkedne el – ez egy 0 és 1 közötti számnak tekinthető – ki kell választani azt az ágat, melynek eltolása a leginkább közelíti ezt a számot (esetleg alkalmazhatunk itt is interpolációt két különböző ág mintái között).

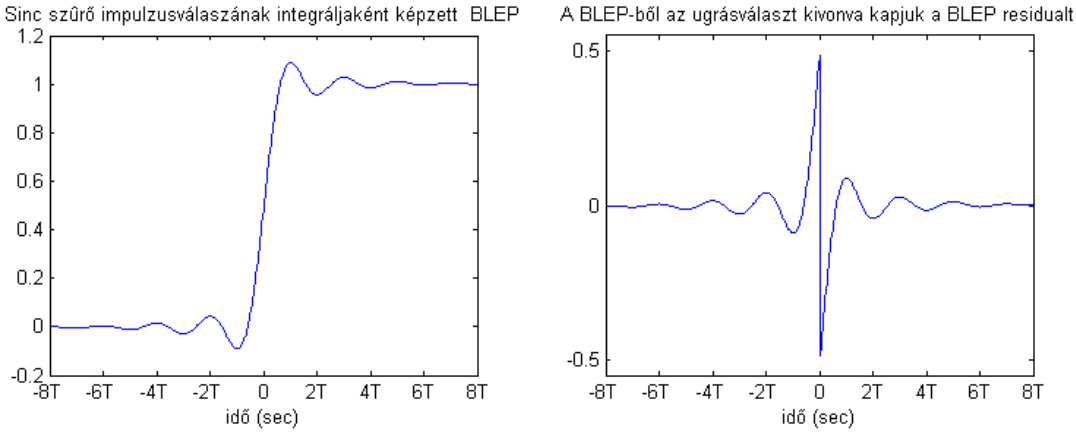
A BLEP módszer

A fűrészjel generálásához szükséges integrálás valós időben zajlik a BLIT módszernél. Hatékonyabbá tehető az algoritmus, ha elkerüljük az integrálás valós idejű elvégzését. Ezt az elvet követi a BLEP módszer (Band-limited step function – sávkorlátozott lépésfüggvény).

A Dirac impulzus integrálja egy egységugrás függvény. Ha ideális impulzus helyett egy sávkorlátozott impulzust – amilyen a sinc impulzus – integrálunk, akkor sávkorlátozott egységugrást kapunk. A BLEP úgy állít elő sávkorlátozott jelet, hogy a triviális módszerrel előállított jelben lévő ugrásokhoz hozzáadja a sávkorlátozott ugrás (ez a BLEP) és az ideális egységugrás különbségét. A különbségképzés eredményét BLEP residualnak (maradék) hívjuk.

Az ideális egységugrás egy diszkontinuitást reprezentál. Egy diszkontinuitás egy jel spektrumát végtelen szélessé alakítja, hiszen egy végtelen gyors változást, ugrást valósít meg. Megvalósítandó jeleink (négyszög, fűrész), melyeket sávkorlátozottá szeretnénk tenni, mind ugrást, diszkontinuitást tartalmaznak. A BLEP módszer ezt a diszkontinuitást teszi sávkorlátozottá. A BLEP residual használatával ily módon bármilyen diszkontinuitást tartalmazó jelet sávkorlátozottá lehet tenni [1]. A BLEP residualt kvázi megfelelő fázisban (és megfelelően skálázva) „rá kell illeszteni” a diszkontinuitásra.

A BLEP-et tehát egy sávkorlátozott impulzust integrálva kaphatjuk meg. Ilyen impulzus a sinc függvény, vagy valamilyen másik aluláteresztő szűrő impulzusválasza. Az integrálást előre elvégezzük (így nem valós időben kell ezt megtennünk – nagyobb hatékonyság), létrehozva egy BLEP táblát, illetve BLEP residual táblát, és mikor diszkontinuitásba „botlunk” a jelben, tartalmát hozzáadjuk ahhoz. Ehhez ki kell tudnunk számítani a diszkontinuitások helyét.



.. ábra A BLEP (balra) és a BLEP residual (jobbra)

Ahogy a BLIT-nél, nyilvánvalóan itt is szükség van a tábláknak különböző fázisban mintavételezett változatait eltárolni, hiszen a diszkontinuitás ritkán esik pont mintára.

Mivel a diszkontinuitás a tábla közepén van, előre kell számolnunk. Ha egy N méretű táblánk van, a diszkontinuitás előtt $N/2$ mintával már ki kell olvasnunk a tábla első elemét, „előre kell néznünk”. Vagy ezt tesszük, vagy késleltetés keletkezik a jelben. Ha ez a késleltetés mindössze pár minta – tehát kisméretű BLEP residualt használunk – akkor az még nem észrevehető,

azonban alacsony mintavételi frekvencia és/vagy nagyméretű tábla esetén ez akar hallható, egyúttal zavaró is lehet.

A 2.7-es ábrán a BLEP függvény és a BLEP residual látható (balra, illetőleg jobbra). Kivehető, hogy a BLEP az egységugrás (mely 0-ban ugrik 1-re) sávkorlátozott közelítése. Ha egy jelünkben előforduló ilyen ugráshoz hozzáadjuk a jobb oldalt látható BLEP residualt – megfelelően skálázva és úgy, hogy annak közepe pontosan az ugrásra essen – akkor azt az ugrást sávkorlátozottá tettük – persze nem teljesen. Hogy mennyire, az az integrált szűrő-impulzusválasztól (tehát a választott szűrőtől) függ. Itt pozitív előjelű az ugrás, ha negatív előjelűről van szó, mint ahogy az emelkedő fűrészjel esetében is, akkor nem hozzáadnunk, hanem kivonnunk kell a BLEP residualt.

A BLEP módszert veszi alapul, de sokkal egyszerűbb algoritmust adó megoldás a PolyBLEP.

A PolyBLEP módszer

A PolyBLEP implementálásához nincs szükség szűrőtervezésre, integrálásra, táblakészítésre. A PolyBLEP szintén azt a koncepciót követi, hogy a hullámforma diszkontinuitása körüli minták módosításával lehetőségünk van a jel sávkorlátozására. Azonban itt a BLEP residualhoz hasonló értékek előállításához egy polinomot használunk (innen a módszer elnevezése). Nem kell egy táblába előre eltárolnunk ezeket az értékeket, mert mindössze a diszkontinuitás előtti és utáni, tehát összesen két mintát módosítunk, és ezeket valós időben számolja az algoritmus. Egy polinomba behelyettesítve a diszkontinuitás helyét a két minta között, megkapjuk azt a két értéket, melyet hozzá kell adnunk a szóban forgó két mintához.

A BLEP módszernél terveznünk kellett egy aluláteresztő szűrőt, s annak impulzusválaszát integrálva kaptuk a sávkorlátozott egységugrást. A PolyBLEP módszer hasonló utat követ, de ezt a sávkorlátozott ugrást zárt képlet formájában írja le, egy polinom formájában. Formálisan itt is integrálásról van szó. Az impulzusválasz, amit alapul vesz, egy háromszög:

$$\begin{aligned} y(x) &= x + 1, ha \ -1 \le x \le 0 \\ y(x) &= 1 - x, ha \quad 0 < x \le 1 \\ &0 \text{ egyébként} \end{aligned} \tag{2.7}$$

Egy ilyen háromszög-impulzusválasszal rendelkező szűrő $sinc^2$ -tel arányos amplitúdó-spektrummal rendelkezik, ami használható antialiasing aluláteresztő szűrőként [1]. A 2.7-es képletben lévő kifejezést integrálva a következőt kapjuk:

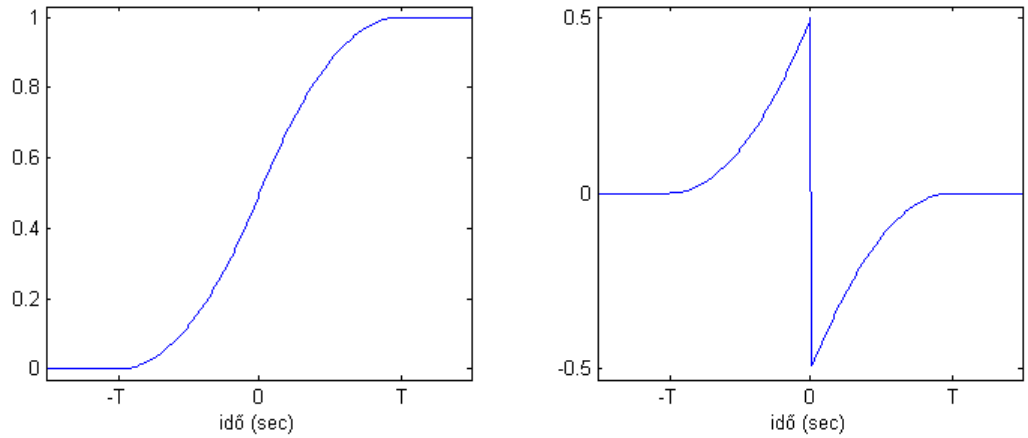
$$\begin{aligned} &0, \text{ ha } x < -1 \\ \int y &= \frac{x^2}{2} + x + C, ha \ -1 \le x \le 0 \\ \int y &= x - \frac{x^2}{2} + C, ha \quad 0 < x \le 1 \\ &1, \text{ ha } 1 < x \end{aligned} \tag{2.8}$$

Ha a 2.8-as képletben szereplő C konstans értékét $1/2$-nek vesszük, akkor a benne szereplő polinomok egy 0 és 1 közötti, BLEP-hez hasonló sávkorlátozott ugrást adnak. Ebből magát az egységugrást kivonva kapjuk a BLEP residualhozhoz hasonló alakú PolyBLEP függvényt:

$$polyblep(x) = \frac{x^2}{2} + x + \frac{1}{2}, ha \; -1 \leq x \leq 0$$
$$polyblep(x) = x - \frac{x^2}{2} - \frac{1}{2}, ha \quad 0 < x \leq 1 \tag{2.9}$$

A 2.9-es képletben szereplő polinomba x helyére a diszkontinuitás előtti és utáni minta diszkontinuitástól való távolságát (mintákban vett távolság) helyettesítve megkapjuk azt a két értéket, amit hozzá kell adnunk ezekhez a mintákhoz (illetőleg kivonnunk, lefelé ugrás esetén, pl. az emelkedő fűrészjelnél).

A PolyBLEP módszer tehát meglehetősen egyszerű, mindössze meg kell határoznunk hozzá a diszkontinuitás helyét, és a polinomba helyettesíteni, majd két minta értékét módosítani. A módszer másodfokú polinomot használ, azonban az elvet követve az eljárás továbbfejleszthető magasabb rendű polinomok előállítására, melyekkel finomabb beavatkozás lehetséges [1].

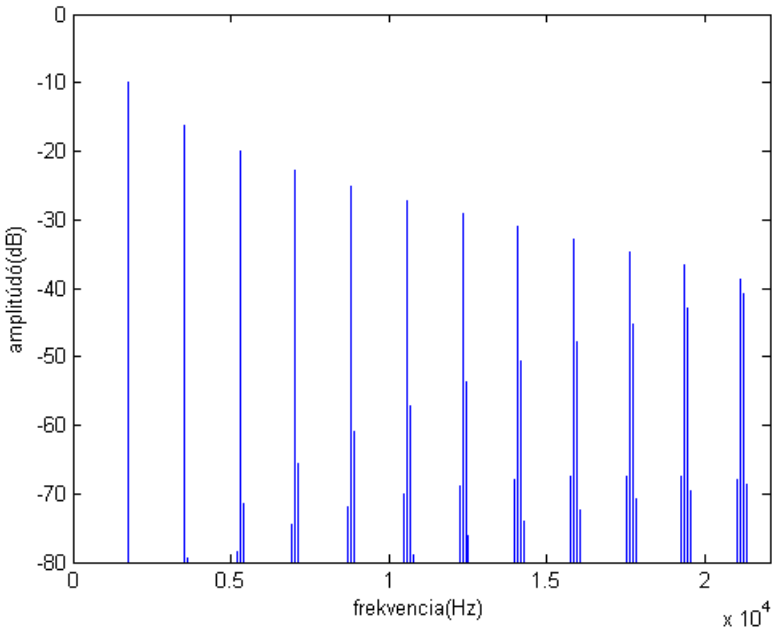


.. ábra A PolyBLEP módszerrel kapott polinomok ábrázolása

A 2.8-as ábrán ábrázoltam a PolyBLEP polinomokat. Az x tengelyre az időt vettem fel, a polinom -1 és 1 határait egy-egy mintányi időbeli távolságnak megfeleltetve. Az ábrán t=0 a diszkontinuitás helye, így szemléletes, hogy az előtte és utána lévő minta helyén lévő értékek lesznek a felhasznált polinom értékek. Az ábrán a –T-n és T-n túli értékek nem a polinom értékei (annak csak -1 és 1 közötti értékeit használjuk), csak a szemléltetés kedvéért szerepelnek.

A PolyBLEP egyszerűségéből is következően nem produkál olyan jó eredményt, mint a BLIT-SWS, vagy a BLEP módszer, de a meghallgatási teszt bizonysága szerint az általa produkált hangminőség kielégítő (az additív szintézissel képzett jeltől szinte nem volt megkülönböztethető, az eltérés pedig nem zaj jellegű, hanem hangszínbeli volt). Egyszerűsége, alacsony számításigénye miatt azok közé az algoritmusok közé tartozik, amik megfelelőek lehetnek az implementálásra a szintetizátorban.

A 2.9-es ábrán a PolyBLEP módszerrel előállított 1760 Hz alapfrekvenciájú fűrészjel frekvenciaképe látható. Az ábrát a 2.2-es ábrával (melyen a triviális megoldással előállított jel spektruma látható) összehasonlítva megállapítható, hogy hatalmas mértékű javulás érhető el a PolyBLEP megoldás alkalmazásával. Az átlapolódott, nemkívánatos komponensek csak 15 kHz fölött közelítik meg a harmonikusok szintjét, s a 2.5-ös ábrával összevetve az is látszik, hogy a harmonikusok amplitúdói is kellően megegyeznek az ideális (additív módszer) esettel 15 kHz alatt. Tudván, hogy az egészen kicsi gyermekektől eltekintve az emberek nagy többsége 15 kHz felett szinte nem is hall, ez nagyon jó eredmény. Ráadásul a belapolódott komponensek a harmonikusok közvetlen közelében vannak frekvenciájukat tekintve[vii], így a frekvenciaelfedési jelenségeknek hála valószínűleg még kevésbé érzékelhetőek a nemkívánatos, zajnak tekintett frekvenciakomponensek.



.. ábra PolyBLEP módszerrel előállított fűrészjel (f=1760 Hz alapfrekvencia) spektruma

[vii] Ez a közelség lebegést produkálhat, mely azonban csak akkor igazán észrevehető, ha a közeli átlapolódott komponens amplitúdójának nagysága megközelíti a harmonikus amplitúdójának nagyságát. Ez azonban csak 15 kHz felett van így, az e fölött lévő komponensek pedig alig (vagy egyáltalán nem) érzékelhetőek az emberi fül számára.

2.5 A spektrum esését módosító megoldások: Lane, DPW, DPW2X

Ebben a fejezetben két olyan megoldással foglalkozom, melyek az eddigiekhez képest eltérő módon közelítik meg a problémát. Először szó esik – a szakirodalomban csak kidolgozójának nevével emlegetett – Lane módszerről, majd a DPW algoritmust mutatom be, végül annak kétszeresen túlmintavételezett változatát.

A Lane módszer

A Lane módszer három lépésben állítja elő a csökkentett alias tartalmú jelet. Első lépésként egyenirányítunk egy szinuszjelet, majd második lépésben egy IIR szűrővel szűrjük azt. Végül a harmadik lépés egy újabb szűrés, ami egy szimpla aluláteresztő szűrő alkalmazását jelenti. A módszer elve szerint az IIR szűrő helyes megválasztásával klasszikus hullámformákat tudunk előállítani.

Szakdolgozatomban a fűrészjel előállításán keresztül mutatom be az egyes megoldásokat. A Lane módszer szerint fűrészjel előállításához a 2.10-es képletben látható IIR szűrőt kell alkalmazni [2]:

$$H(z) = \frac{\alpha(1 - z^{-1})}{1 - \gamma z^{-1}}$$

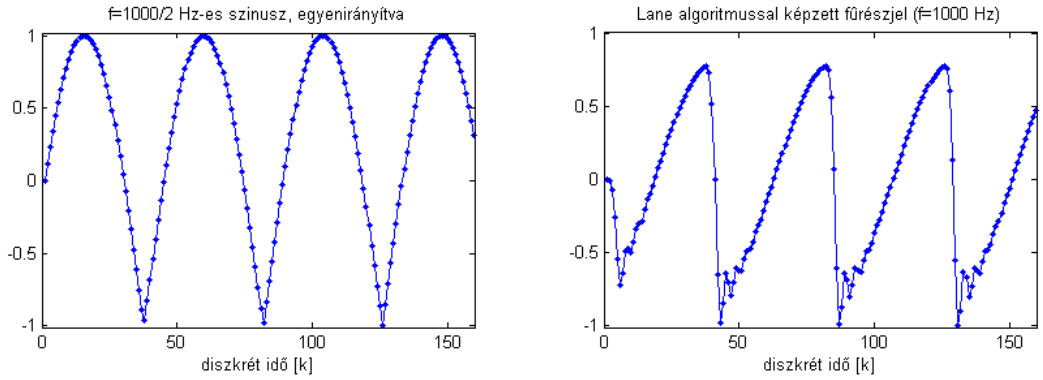
$$\gamma = \frac{\cos(\omega_c)}{[1 + \sin(\omega_c)]} \tag{2.10}$$

$$\alpha = \frac{1 + \gamma}{2}$$

$$\omega_c = \frac{2\pi 16 f}{f_s}$$

ahol $H(z)$ a szűrő átviteli függvénye, alatta pedig a paraméterek képletei láthatóak. Szokás szerint f_s jelöli a mintavételi frekvenciát, f a jel kívánt alapfrekvenciája.

A harmadik lépésben használt szűrő egy aluláteresztő szűrő. Nincs rá speciális kikötés, Lane és társai Butterworth szűrőt használtak, különböző vágási frekvenciákat és fokszámokat próbáltak ki. Ez az utolsó szűrés a magas frekvenciás alias komponensek kiszűrésére szolgál, ugyanakkor az átlapolódott komponensekkel együtt nyilván a harmonikusokat is kiszűri. Utóbbi miatt használatának jogossága megkérdőjelezhető, hiszen ugyan eltüntetjük vele magas frekvencián az átlapolódott, zajnak tekintett komponenseket, de a kívánt harmonikusokat is, így pedig nem tekinthető a jel az eredeti jel hű szimulációjának.

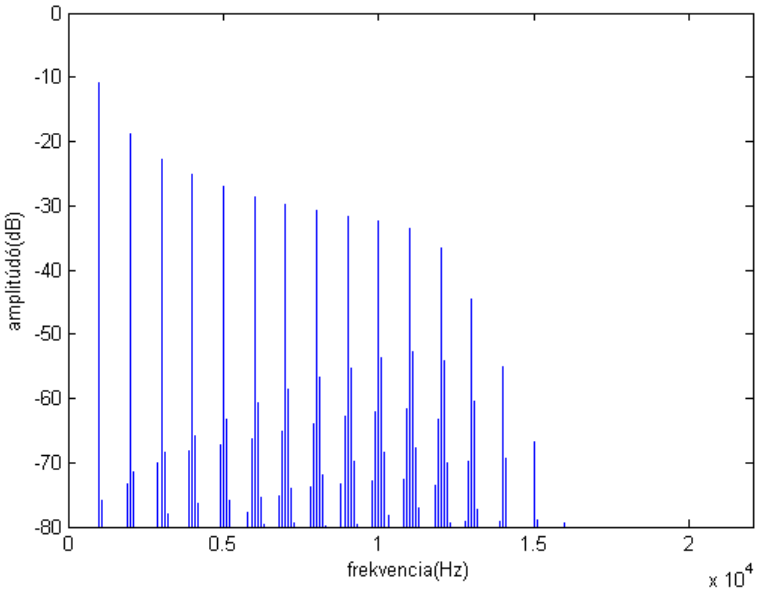


.. ábra Lane algoritmus: balra az egyenirányított szinuszjel, jobbra a szűrők alkalmazása után kapott fűrészjel

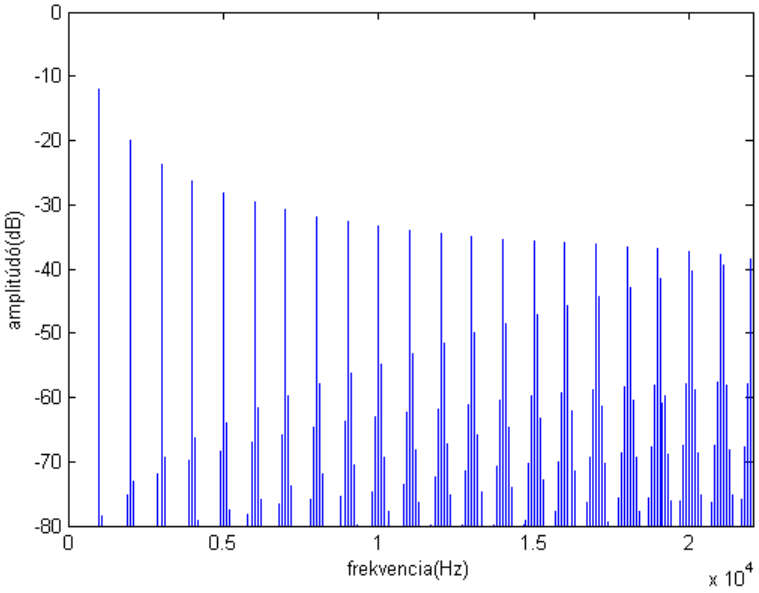
A 2.10-es ábrán láthatjuk az egyenirányított szinuszjel és a Lane algoritmussal előállított fűrészjel első pár periódusát. Jól látható, hogy az ideális fűrészjel alakjához képest jelentésen eltérő a kapott hullámforma. Érdemes megjegyezni, hogy az egyenirányítandó szinuszjel frekvenciája fele kell, hogy legyen a fűrészjel kívánt alapfrekvenciájának, ahogy ez az ábrán is jól látszik. Az ábra előállításakor használt aluláteresztő szűrő (a harmadik lépés) egy a Matlab „butter” parancsával tervezett nyolcadrendű Butterworth szűrő volt, 12 kHz vágási frekvenciával.

A 2.11-es ábrán látható a Lane módszerrel előállított fűrészjel (az alapharmonikus 1000 Hz) frekvenciaképe. Az átlapolódott komponensek jóval kevésbé vannak jelen, mint a triviális megoldásnál, ugyanakkor megfigyelhető, hogy az előző fejezetben bemutatott PolyBlep módszer jobb eredményt ad (szemre legalábbis biztosan; bővebben egy későbbi pontban hasonlítom össze a kipróbált módszereket). A harmonikusok egészen 10 kHz-ig jól követik az ideális értékeket, azonban utána a harmadik lépésnek köszönhetően jelentősen lecsökkenek. 15 kHz felett már egyáltalán nem észlelhetőek komponensek (se kívánt, se alias).

A 2.12-es ábrán ábrázoltam a harmadik lépés kihagyásával készült jel spektrumát is, így megfigyelhető annak jelre gyakorolt hatása.



.. ábra Lane algoritmussal előállított fűrészjel spektruma (f=1000 Hz alapfrekvencia)



.. ábra Lane algoritmussal előállított fűrészjel spektruma (f=1000 Hz alapfrekvencia), ha kihagyjuk a harmadik lépést (aluláteresztő szűrés)

Fontos megemlíteni, hogy a 2.10-es képletben megadott IIR szűrő az alapfrekvencia csak egy bizonyos tartományában teszi lehetővé a fűrészjelek előállítását. A szűrő ugyanis nem stabil abban a teljes tartományban, amelyben az alapfrekvenciák lehetnek a szintetizátor (zongora) billentyűi szerint.

A 2.10-es képletből leolvasható, hogy a szűrő pólusa γ. Ez a pólus az alapfrekvenciától függő. Tudjuk, hogy egy diszkrét idejű rendszer akkor gerjesztés-válasz stabilis, ha pólusai az egységkörön belül találhatóak. A γ-t leíró képletből kiderül, hogy γ – így a szűrő pólusa – a

valós tengelyen van, valós értékeket vehet fel. Akkor lesz kisebb 1-nél, ha ω_c 0 és π közé esik. A ω_c-t megadó képletből pedig kiszámolható, hogy ekkor f-nek 0 Hz és 1378.125 Hz között kell lennie. Egész pontosan a $2k\pi$ és $(2k+1)\pi$ tartományban kell ω_c-nek lennie, és ennek megfelelőek az f -re vonatkozó megkötések is. A fentiek miatt ábrázoltam f=1000 Hz alapfrekvenciájú jeleket a Lane módszer ábráihoz a szokásos, 1760 Hz-es „A" hang helyett. E korlátok miatt sajnos a vázolt algoritmus alkalmatlanná válik a szintetizátorban való implementációra. Ugyanakkor, mint később ki fog derülni, nem szenvedünk komoly hátrányt ebből következően, ugyanis a Lane módszer nem a legjobb megoldás a problémánkra.[viii]

A DPW módszer

Az utolsó tárgyalt megoldás szakdolgozatomban a DPW (Differentiated Parabolic Wave – differenciált parabolikus hullám) módszer, és annak kétszeresen túlmintavételezett változata.

A DPW algoritmus arra a megfigyelésre alapul [2], hogy egy szakaszosan parabolikus hullám differenciálásával a fűrészjel jó közelítését kaphatjuk meg.[ix]

Az ideális, folytonos idejű fűrészjel harmonikusai 6 dB/oktáv sebességgel csökkennek a frekvencia növekedtével. A DPW algoritmus oly módon javít az aliasing hibán, hogy nem közvetlenül a fűrészjel mintavételezett változatát állítja elő (ahogy azt a triviális megoldás teszi), hanem annak integrálját, ami szakaszosan parabolikus hullám. A parabolikus hullám előállítása után (tehát a mintavételezés pillanata után) differenciálással állítja vissza a fűrészjel formát. A szakaszosan parabolikus hullám frekvenciakomponensei 12 dB/oktáv sebességgel csökkennek a frekvencia növekedtével. Így ha ennek állítjuk elő mintavételezett verzióját (diszkrét idejű reprezentációját), akkor jóval alacsonyabb lesz az átlapolódott, alias komponensek szintje, mint a triviális megoldásnál.

A DPW négy lépésben állítja elő a fűrészjelet. Első lépésként generáljuk a triviális fűrészjelet, második lépésben négyzetre emeljük azt, így kapjuk meg a parabolikus hullámot[x]. A harmadik lépésben differenciáljuk a jelet, visszanyerve a fűrész hullámformát. Végül a negyedik lépés a jel skálázása (szorzás egy alapfrekvenciától függő számmal), ami bármilyen alapfrekvencia esetén -1 és 1 közé skálázza a jelet. Az, hogy milyen megoldást használunk a differenciáláshoz, szabadságot ad különféle DPW verziók előállítására.

A legegyszerűbb megoldás a differenciálásra a 2.11-es képletben szereplő elsőrendű FIR differenciáló. Kimenete a bemenet aktuális és azt megelőző értékének a különbsége.

[viii] Hozzá kell tenni: bizonyos, hogy a módszer kiötlői orvosolták a fenti problémát, azonban a Lane módszer teljes dokumentációját sajnos nem tudtam fellelni, mindössze egy kivonatát, s az alapján próbáltam ki.

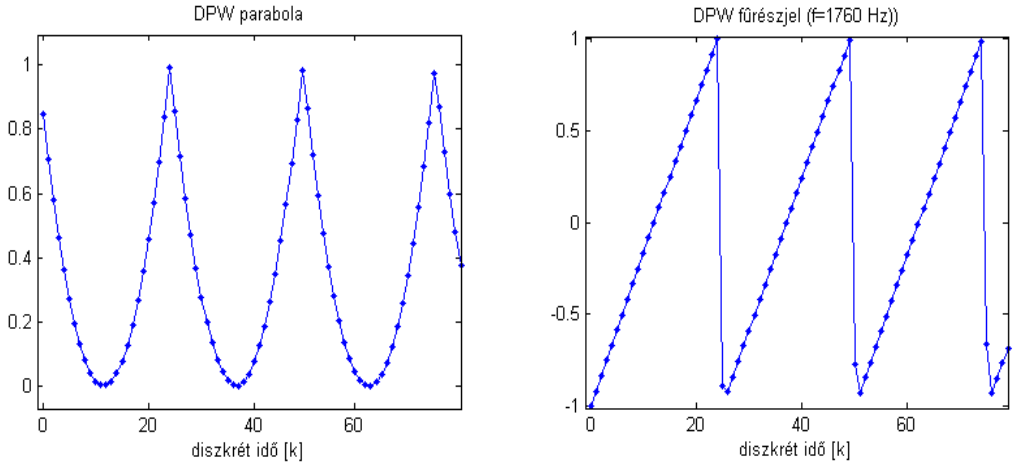
[ix] A Lane módszer is hasonló elven működött. Ott a szakaszosan parabolikus hullámhoz igen hasonló egyenirányított szinusz differenciálása adta a fűrészjelet.

[x] A fűrészjel szakaszosan lineáris függvény: $y = x$, integrálja szakaszosan parabola: $\int y = \frac{x^2}{2}$, innen „jön" a négyzetre emelés.

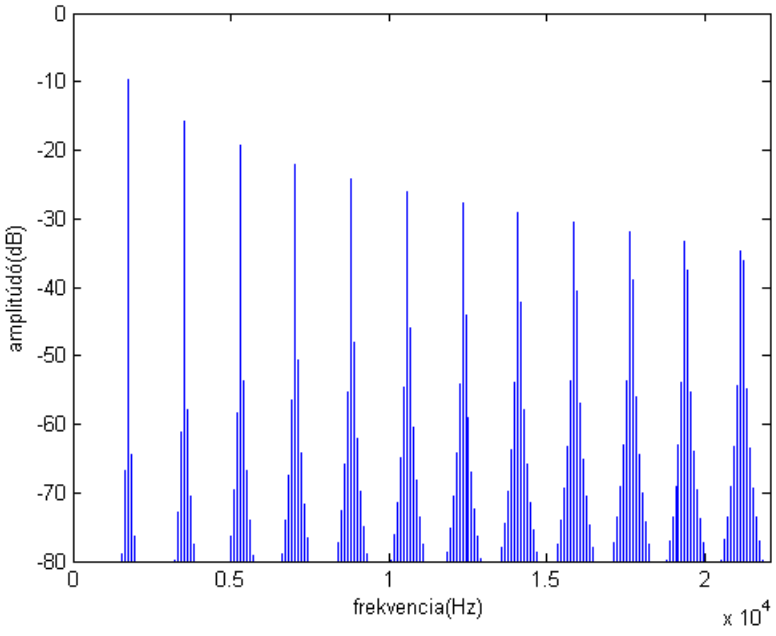
$$H_{diff}(z) = 1 - z^{-1} \tag{2.11}$$

$$c = \frac{f_s}{\left[4f\left(1 - \frac{f}{f_s}\right)\right]} \tag{2.12}$$

A 2.12-es képletben szereplő c a skálázó faktor, amellyel utolsó lépésként megszorozzuk a jelet, s ami a jelet -1 és 1 közé szorítja. Itt a lényeg nyilvánvalóan nem az amplitúdó nagysága, hanem a tény, hogy alapfrekvenciától függetlenül azonos amplitúdójú jelet kapunk. A képletben f jelöli az alapfrekvenciát, f_s a mintavételi frekvenciát.



.. ábra DPW: balra a szakaszosan parabola hullám, jobbra a differenciálásával kapott fűrészjel



.. ábra DPW algoritmussal előállított fűrészjel (f=1760 Hz alapfrekvencia) spektruma

A 2.13-as ábrán egy 1760 Hz-es alapfrekvenciájú DPW algoritmussal készült fűrészjel első pár periódusa látható. Egy -1 és 1 között futó számláló kimenetét négyzetre emelve kapjuk a bal oldali ábrán látható szakaszosan parabolikus hullámot. A fent leírt differenciálással (és skálázással) jutunk a jobb oldalt látható fűrész hullámhoz.[xi]

A 2.14-es ábrán a jel frekvenciaképét ábrázoltam. Az alias komponensek körülbelül azon a szinten vannak, mint a Lane módszernél (a DPW azonban kisebb számításigényű). A triviális módszerrel összehasonlítva látható, hogy jelentősen lecsökkentette a hibát az algoritmus. Ugyanakkor a PolyBLEP-pel való összevetésben alulmarad a DPW spektruma.

A DPW túlmintavételezése

Korábban, a 2.2-es fejezetben említettem, hogy az alias hiba csökkentésének kézenfekvő módja a jel túlmintavételezése. A következőkben a DPW algoritmus kétszeresen túlmintavételezett verziójának bemutatásával illusztrálom ezt a módszert.

A 2.2-es fejezetben, az aliasing probléma bemutatásánál láttuk, hogy a mintavételi frekvencia nagysága határozza meg az átlapolódás nagyságát. Sávkorlátozott jelek mintavételezésénél a Nyquist-kritérium betartásával, elegendően nagy mintavételi frekvencia választásával teljesen elkerülhető az aliasing hiba. Az általunk előállítani kívánt hullámformák azonban, mint tudjuk, nem sávkorlátozottak, ezért kellett más megoldásokhoz folyamodnunk. Nagyobb mintavételi frekvencia választása azonban ettől még természetesen csökkenti a hibát.

Szoftveres szintetizátorom megvalósításánál egy adott mintavételi frekvenciájú jelet kell előállítanom (44.1 kHz, amellyel a hangkártya dolgozik). Ebből kifolyólag, ha ettől eltérő, nagyobb f_s-t alkalmazunk, akkor azt a folyamatban csak ideiglenesen tehetem, később vissza kell térnünk az eredeti mintavételi frekvenciára.

A DPW algoritmus bonyolultsága, számítási igénye kicsi, ezért jöhet szóba a túlmintavételezés, ami egyébként elég gazdaságtalan módszer, hiszen ahányszoros a túlmintavételezés, egy kimeneti mintához annyi mintát kell előzetesen előállítanunk. Túlmintavételezni csak az eredeti f_s egész számú többszörösein hatékony, hiszen ekkor a folyamat végén lévő alulmintavételezés során nem kell interpolálnunk (minden n-edik mintát vesszük). Ha interpolációra kényszerülnénk, az feleslegesen komplexszé tenné az algoritmust.

A módszer három lépésből áll. Elsőként létrehozzuk a kívánt jelet a magasabb mintavételi frekvencián (túlmintavételezés), második lépésben aluláteresztő, decimáló szűrővel szűrjük, végül pedig visszatérünk az eredeti f_s-hez (alulmintavételezés). Az utóbbi két lépés együttesen a decimálás. A magasabb mintavételi frekvencián létrehozott jel szűrése (decimáló szűrő) antialiasing szűrésnek felel meg, pontosan úgy, mint mikor egy folytonos jelet szűrünk mintavételezés előtt (az $f_s/2$ feletti komponensek eltávolítása).

Ezt a koncepciót a DPW algoritmusnál a következőképp valósítom meg. A szakaszosan parabolikus jelet kétszeres mintavételi frekvencián generálom (DPW2X), az

[xi] Megjegyzendő, hogy az ábrán látottakkal ellentétben célszerű nem -1, hanem 0 értékről indítani a jelet, hogy elkerüljük az ilyenkor hallható „pattanást” lejátszáskor. Ehhez mindössze a számláló kezdőfázisát kell megfelelően beállítanunk.

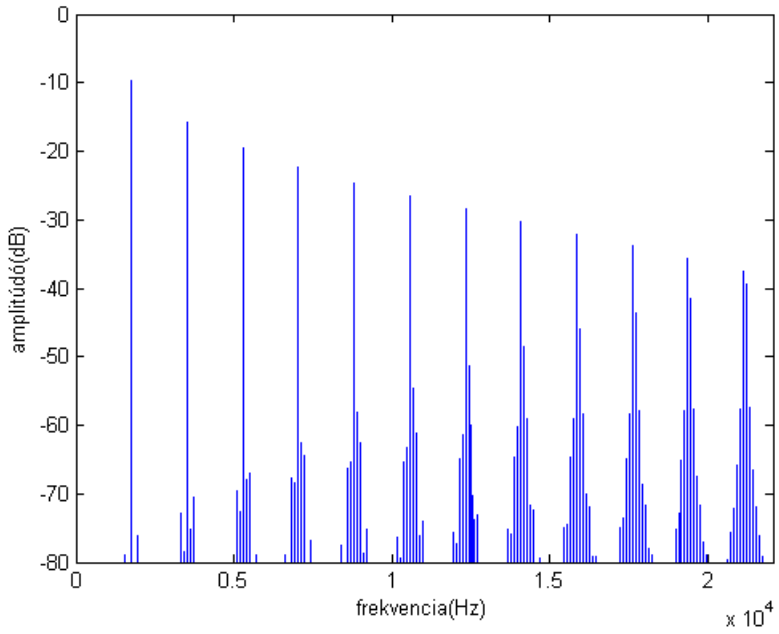
alulmintavételezéssel egy lépésben alkalmazom a decimáló szűrőt, majd a differenciálással megkapom a fűrészjelet. A decimáló szűrő a következő:

$$H_{dec}(z) = \frac{1 + z^{-1}}{2} \tag{2.13}$$

Ez egy elsőrendű mozgó átlag FIR szűrőnek felel meg, ami a legegyszerűbb megoldásnak tekinthető. A decimáló szűrés utáni alulmintavételezés során minden második mintát veszem a jelből, így visszakapom az eredeti mintavételi frekvenciát. A decimálás két művelete egy lépésben is elvégezhető, ha az átlagolást csak minden második szomszédos mintapárra alkalmazom, s csak az átlagolások eredményeit veszem (ahelyett, hogy minden egymást követő mintát átlagolnék, s csak ez után választanám ki minden másodikat). A 2.13-as decimáló szűrő a kétszeres f_s felénél, tehát a végső mintavételi frekvenciánál nyom el a legjobban, így azokat a komponenseket csillapítja jól, melyek később alacsony frekvencián jelennének meg belapolódott hibaként. Vagyis azoknak az alias komponenseknek a megjelenését előzi meg, amelyek a leghallhatóbbak lennének. Természetesen másfajta decimáló szűrő is megvalósítható. A DPW2X algoritmusban a decimáló és a differenciáló tagok különböző megvalósításaival lehetőségünk van optimalizálni a megfelelő számítási hatékonyság és a kívánt minőség tekintetében.

A kétszeres túlmintavételezéssel előállított DPW fűrészjel spektrumát a 2.15-ös ábrán tekinthetjük meg (f=1760 Hz alapfrekvencia). Érdemes összehasonlítanunk a 2.14-es ábrával, hogy a túlmintavételezés jótékony hatását megfigyelhessük. Szembetűnő a javulás az eredeti DPW algoritmushoz képest, a triviális megoldáshoz képest pedig főképp alacsony az átlapolódott komponensek intenzitása. A 2.9-es ábrán ábrázolt PolyBLEP féle spektrummal összevetve látszik azonban, hogy még a DPW kétszeres túlmintavételezése sem hoz olyan jó eredményt, mint a PolyBLEP.

Az algoritmusok Matlab-ban történő implementációja utáni meghallgatási teszt során megállapítottam, hogy a legutóbb ismertetett módszer, a DPW2X, valamint a PolyBLEP nyújtott olyan hangminőséget, melynél az aliasing jelenséget nem észleltem. A módszerek részletesebb, precízebb összehasonlítása a következő fejezet tárgya.



.. ábra DPW2X algoritmussal előállított fűrészjel (f=1760 Hz alapfrekvencia) spektruma

2.6 A tárgyalt oszcillátor algoritmusok összehasonlítása

Az eddigiekben áttekintettem a fellelhető megoldásokat, ebben a fejezetben összehasonlítom őket, hogy kiderüljön, melyiket, s mi alapján ítélem a legalkalmasabbnak az implementációra.

Az összehasonlítás kritériumai

Az összehasonlítás alapja egy egyszerű kritériumrendszer. A 2.1-es fejezetben említetteknek megfelelően ez négy elemből áll.

A két legfontosabb szempont egyike a számítási igény, az algoritmus egyszerűsége. Itt arra próbálok becslést adni, hogy egy kimeneti minta (fűrészjel generálásakor) előállításához mennyi elemi műveletre van szükség. A második szempont, s szintén a két legfontosabb egyike, a meghallgatási teszt. Olyan algoritmust választok, ami megfelel a saját – nyilván szubjektív – igényemnek. A Matlab-ban implementáltam az összevetni kívánt algoritmusokat, mindegyiket több alapfrekvencián hallgattam meg.[xii]

A harmadik kritérium az algoritmusok jel/zaj viszonyának összehasonlítása több alapfrekvencián. A jel/zaj viszonyt a következőképpen számoltam. Jelnek minden algoritmus jel/zaj viszonyánál az ideálisan sávkorlátozott megoldás, az additív szintézis által adott harmonikusokat vettem. Az adott algoritmus zaja pedig a harmonikusokon kívüli komponensek összessége. Mindehhez a Matlab fft parancsát használtam, a jel és zaj energiák számításához az fft eredményvektorának elemeit négyzetesen összegeztem. Végül a jel és a zaj energiájának hányadosát decibelre váltottam, így alakult ki a végső SNR (Signal-to-noise) érték. Meg kell jegyezni, hogy csak olyan alapfrekvenciákon vizsgáltam az SNR-t, ahol f nem osztója f_s-nek.

[xii] A CD-mellékleten található program (SawOsc.exe) segítségével az olvasó is meghallgathatja az algoritmusok eredményeit.

Így azokat az eseteket nem vettem számításba, ahol az átlapolódott komponensek az eredeti harmonikusokra „esnek”. Ezekben az esetekben zaj jellegű hiba nincs, ellenben amplitúdóhibát kapunk. Ezek az esetek azonban nem érintik a szintetizátor megvalósítását, ugyanis a zongorán (s ebből következően a MIDI billentyűzeten) mindösszesen egy hang, a 196 Hz-es „G” hang olyan, melynek frekvenciája osztója a mintavételi frekvenciaként a szoftveres szintiben használt 44.1 kHz-nek [5].

A negyedik, legkevésbé lényeges szempont az algoritmus kezelhetősége a vibrato, vagyis az alapfrekvencia folytonos változtatása szempontjából.

Számításigény

Elsőként az additív szintézis nyújtotta megoldással foglalkoztam. Ennek számítási igénye jelentős, hiszen minden egyes mintához annyi szinuszos tagot kell összegezni, ahány harmonikus a Nyquist-frekvencia alá „befér”. (Ez egy 220 Hz-es „A” hang esetén pl. 100 harmonikus számítását jelenti.) N darab harmonikus esetén N-1 összeadás és N darab szinuszos oszcillátor érték számítását jelenti (ez utóbbi önmagában is bonyolultabb, mint egy összeadás). Az additív szintézist ezért csak nagy alapfrekvenciájú jelek számításához érdemes használni, ahol olyan kevés harmonikust kell számítani, hogy esetleg hatékonyabb, mint az egyébként használt algoritmus.

A DPW algoritmus számításigénye kicsi. Egy-egy minta számításához léptetnünk kell egy számlálót (egy összeadás), a négyzetre emelés egy szorzást igényel. A differenciálás szintén egy összeadás (kivonás, de számítástechnikai értelemben véve összeadásnak számít). Végül a jel skálázása ismét egy szorzást von maga után. Ez összesen 2-2 összeadás és szorzás, esetleg egy komparálás a számláló túlcsordulását figyelendő. Kijelenthető, hogy ez a módszer gazdaságos, a triviális megoldáshoz (amely mindössze egy összeadást igényel mintánként) a legközelebb áll ebben a tekintetben.

A DPW kétszeresen túlmintavételezett változata, a DPW2X négy összeadást és három szorzást követel meg mintánként. Mivel kétszeres mintavételi frekvencián hozzuk létre a szakaszosan parabolikus hullámot, ezért egy kimeneti mintához két összeadás tartozik a számlálóban. A parabolához ugyanúgy két mintát emelünk négyzetre egy kimeneti mintához, ez kettő szorzás. A decimáló szűrő és a differenciálás egy-egy összeadás, s végül itt is megtalálható a skálázás – megint egy szorzás. Ez a mennyiség még mindig gazdaságosnak számít, a DPW számításigényének kevesebb, mint kétszerese.

A PolyBLEP algoritmus az additív szintézishez hasonlóan nem állandó számítási igényű. A diszkontinuitásokkal nem szomszédos minták számításához elég egy összeadás (számláló léptetése). Több művelet csak a fűrészjel ugrásaival szomszédos két minta kalkulálásához szükséges. Minél magasabb az alapfrekvencia, annál kevesebb mintából áll egy periódus, annál több ilyen „szomszédos” minta van. E két mintához, a két PolyBLEP polinom értékeinek számításához két-két összeadás, egy-egy szorzás (a négyzetes tagok) kell, s az értékeket hozzáadjuk a szomszédos mintákhoz – újabb két összeadás. Egy diszkontinuitással szomszédos minta számításigénye tehát négy összeadás, egy szorzás. A diszkontinuitások helyének megállapítása szintén számítást von maga után. Lehetséges, hogy a hang előállítása előtt

meghatározzuk a (mintákban vett) periódusidőt (többnyire tört szám). Így az ugrás helyét megkapjuk, ha hozzáadjuk az előző ugrás helyéhez ezt az értéket. Ez jelperiódusonként egy összeadás. Ha a frekvenciát folyamatosan változtatjuk (vibrato), akkor erre nincs mód, ekkor pl. egy komparátorral figyelhetjük (feltétel vizsgálat), hogy ugráshoz értünk-e. Ha igen, a diszkontinuitás utáni mintából számítható az ugrás mintaközi helye egy szorzással. Az implementációhoz ez utóbbi esetet kell figyelembe venni.

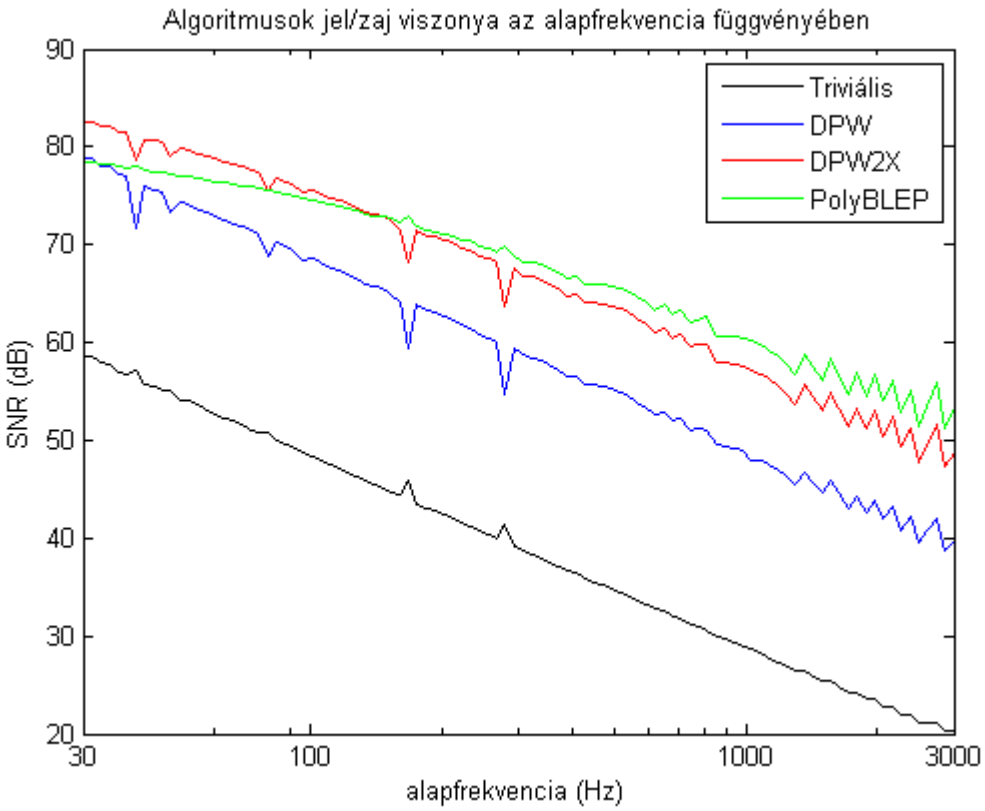
A vizsgált algoritmusok számításigényét a 2-1. táblázatban foglaltam össze:

Algoritmus neve	Egy minta számításához szükséges műveletek
Triviális	1 összeadás
Additív (N darab harmonikus)	N-1 összeadás, N db szinuszos oszcillátor
DPW	2 összeadás, 2 szorzás
DPW2X	4 összeadás, 3 szorzás
PolyBLEP	Ugrással nem szomszédos minta: 1 összeadás Ugrással szomszédos minta: 4 összeadás, 1 szorzás Ugrás helyének számítása: komparálás minden mintánál, periódusonként 1 szorzás

2-. táblázat Oszcillátor algoritmusok számításigénye

Megállapítható, hogy a DPW, DPW2X, valamint PolyBLEP módszerek nagyságrendileg hasonló mennyiségű számítást igényelnek, s ez a mennyiség elég alacsony, mintánként pár összeadás, valamint szorzás. A PolyBLEP tekinthető a leggazdaságosabbnak. Ugyan az ugrások körüli minták számításigénye körülbelül kétszerese a DPW2X algoritmusénak, de ez egy periódusban csak két mintát érint. A legmagasabb, 108-as számú MIDI[xiii] hangjegy frekvenciája 4186 Hz [5], ez 44.1 kHz-es mintavételi frekvenciánál 10.5351 minta hosszú periódust jelent. Mindez azt jelenti, hogy ha 1 mintára átlagolunk, még a legmagasabb hangnál is 10 felé oszlik a diszkontinuitás környéki számítástöbblet.

[xiii] A MIDI-ről egy későbbi fejezetben esik szó.



.. ábra Oszcillátor algoritmusok jel/zaj viszonya

Jel/zaj viszony

A 2.16-os ábra mutatja a korábban leírt jel/zaj viszony számítások eredményeit. A vizsgált alapfrekvenciákat logaritmikus léptékben vettem fel, követve a zenei hangok frekvenciáinak menetét.

Szembeötlő, hogy minden ábrázolt algoritmus jel/zaj viszonya a nagyobb alapfrekvenciák felé egyre rosszabb. A 2.2-es fejezetben említetteknek megfelelően, minél közelebb van az alapfrekvencia a mintavételi frekvenciához, annál nagyobb az átlapolódott komponensek száma, energiája. Az is megfigyelhető, hogy az algoritmusok nagyjából logaritmikusan romlanak f növekedtével (az ábrán ez lineárisnak látszik, hiszen az x tengely logaritmikusan lett felvéve). Mindez azt is jelenti, hogy az algoritmusok egymáshoz képesti viszonya hasonló minden frekvencián (a PolyBLEP kivételével).

Jól látható, hogy a triviális megoldás – természetesen – mind közül a leggyengébb. Ábrázolásával viszonyítási alapot kapunk a többi algoritmushoz. A DPW módszer durván 20 dB javulást produkál a triviálishoz képest. A kétszeres túlmintavételezéssel pedig körülbelül további 8-10 dB minőségemelkedés érhető el.

A legjobb minőséget a PolyBLEP módszer adja (ezt vártuk a korábbi fejtegetések alapján is), az egészen alacsony frekvenciáktól eltekintve. Nagyjából 150 Hz-ig a DPW2X hozza a legmagasabb jel/zaj viszonyt, innentől azonban a PolyBLEP jobb, kb. 3 decibellel (valószínűsíthetően ez a különbség nem, vagy alig érzékelhető, olyan kicsi).

Bár nem a teljes tartományban mutatja a legjobb értékeket, mégis egyértelműen a PolyBLEP mondható a legjobbnak. Egyrészt mert a 150 Hz alatti alapfrekvenciájú jeleknél a legkisebb az aliasing mértéke, ezeken a frekvenciákon még a triviális megoldás is viszonylag elfogadható hangminőséget produkál. Másrészt az emberi hallás érzékenysége frekvenciafüggő, és a legnagyobb érzékenységet a középfrekvenciákon mutatja (kb. 1 és 3 kHz között), alacsony és magas frekvenciákon kevésbé érzékeny. Mindez megmutatja a 2.16-os ábra hiányosságát: a jel/zaj viszony értékek számításánál az igazán pontos eredmény eléréséhez súlyozni kell a komponenseket az emberi hallás érzékenységét figyelembe véve. Tovább pontosíthatunk az algoritmusok keltette hibaérzet mérésén, ha a frekvenciaelfedési jelenségeket is figyelembe vesszük.

Meghallgatásos teszt

Az algoritmusokat Matlab-ban implementáltam, majd mindegyikből egy másodperc hosszú (44100 minta) jeleket generáltam különböző alapfrekvenciákon (30 Hz-től 3 kHz-ig). E mintákat meghallgatva megállapítottam, hogy a DPW algoritmusnál még meghallom az alias zajokat, a DPW2X és PolyBLEB módszerekkel generált jeleknél azonban nem. Utóbbi kettő között különbséget nem véltem felfedezni. A DPW2X és a PolyBLEP kielégítő hangzást nyújtott, annak ellenére, hogy az additív szintézissel képzett jeltől meg tudtam különböztetni őket (bár ez az érzékelésbeli különbség olyan kicsi volt, hogy nem lehet kizárni az érzéki csalódást sem). A különbség azonban nem zaj jellegű volt, inkább talán minimális hangszínbeli eltérést tapasztaltam a magas frekvenciákon.

Modulációra való érzékenység

Az utolsó kritérium a frekvenciamodulációra való alkalmasság. Itt egyáltalán nincs különbség az itt összehasonlított módszerek között, a DPW (és természetesen a DPW2X) és a PolyBLEP esetén is a számláló lépésközének változtatásával tehetjük ezt meg. Mindez egyszerűen kivitelezhető, akár mintánként változtatható, így nem okoz bonyolítást az algoritmusokban.

Végeredmény

A módszerek összehasonlítása a PolyBLEP „győzelmével” zárult. Annak ellenére, hogy nem hallottam ki a minőségbeli fölényét a DPW2X-el szemben, nem zárható ki, hogy más számára érzékelhető az SNR-ben mutatkozó többlet. Ezen kívül számítási hatékonyságban is a legjobb, habár a DPW-k itt sem maradnak le. A DPW2X módszert egyformán hatékonynak ítélem meg az aliasing probléma kezelésében, az implementációmban azonban a PolyBLEP-et alkalmazom.

2.7 Egyéb hullámformák előállításáról

Az oszcillátor algoritmusokat fűrészjelek előállításán keresztül mutattam be. A szubtraktív szintézis technikában jellemzően a fűrészen kívül két másik, felharmonikusokban gazdag hullámformát használnak. Ezek a négyszög- és háromszög hullámformák. Ebben a fejezetben röviden bemutatok néhányat generálásuk módjai közül.

Négyszögjel előállítása

Aliasmentes négyszögjelet hozhatunk létre sávkorlátozott fűrészjelek felhasználásával. Két, egymáshoz képest fázisban eltolt fűrészjel különbsége négyszögjel lesz. A fáziskülönbség megválasztásával kontrollálni tudjuk a különbségképzés eredményeként kapott négyszögjel kitöltési tényezőjét. Ha az egyik fűrészjel pontosan fél periódussal siet/késik a másikhoz képest, akkor a négyszögjel kitöltési tényezője 50% lesz. Ennél kisebb vagy nagyobb fáziskülönbség eltérő kitöltési tényezőt eredményez.[xiv] Figyelnünk kell arra, hogy a fűrészjelek különbségeként létrehozott, 50 %-ostól eltérő kitöltési tényezőjű négyszögjelek DC komponenst tartalmaznak. Csúcstól csúcsig vett amplitúdójuk megegyezik a fűrészjelekével, azonban a DC komponens értékével eltolódnak. Így ha a fűrészjelünk -1 és 1 közötti értékeket vett fel (a hangkártya ebben az értéktartományban várja a mintákat), az 50 %-tól eltérő kitöltési tényezőjű négyszögjelre ez nem lesz igaz. Jelünket ezért skáláznunk kell, hogy a (-1,1) tartományba essen.

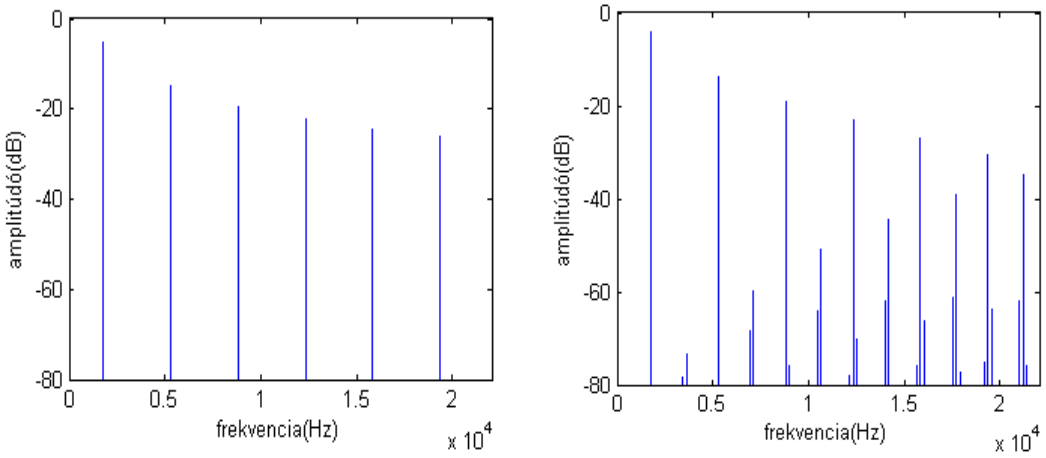
Két fűrészjel különbségének képzésével előállított négyszögjel számítási igénye kétszerese a fűrészjel számítási igényének. Előfordulhat, hogy gazdaságosabb megoldásra van szükségünk.

Egy késleltetővonal beiktatásával egy fűrészjelből, önmaga saját késleltetett verziójának kivonásával is számíthatunk négyszögjelet. Mivel azonban egy késleltetővonallal egész számú mintányi késleltetést tudunk csak megvalósítani, nem tudunk bármilyen frekvenciájú jelet létrehozni. A probléma kiküszöböléséhez törtrész késleltetést kell megvalósítanunk, pl. interpolációval. Ez azonban többletszámítást von maga után, aminek köszönhetően ez a módszer nem lesz hatékonyabb a két fűrészjelet alkalmazónál.

A PolyBLEP módszerrel közvetlenül is létrehozhatunk négyszögjelet. Ugyanazt az elvet kell követnünk, mint fűrészjel generálásakor: a triviális módszerrel generált jelben a diszkontinuitásokkal szomszédos mintákat simítanunk kell a PolyBLEP polinom értékeivel. Emelkedő ugrás esetén hozzáadnunk, negatív irányú ugrás esetén kivonnunk kell a diszkontinuitás előtti és utáni mintából a kapott értékeket. Négyszögjel előállításakor a pozitív élhez érve hozzáadnunk, negatív élhez érve kivonnunk kell.

A 2.17-es ábrán az ideális (additív szintézissel alkotott), valamint a PolyBLEP algoritmussal képzett négyszögjel spektrumát ábrázoltam.

[xiv] A kitöltési tényezőt jelen esetben elég 0 és 50 % között értelmeznünk. A 10 %-os és a 90 %-os kitöltési tényezőjű négyszögjel csak előjelében különbözik egymástól, egymás ellentettei. Így spektrumuk nem különbözik, a hallgató számára ugyanazt az érzetet keltik.



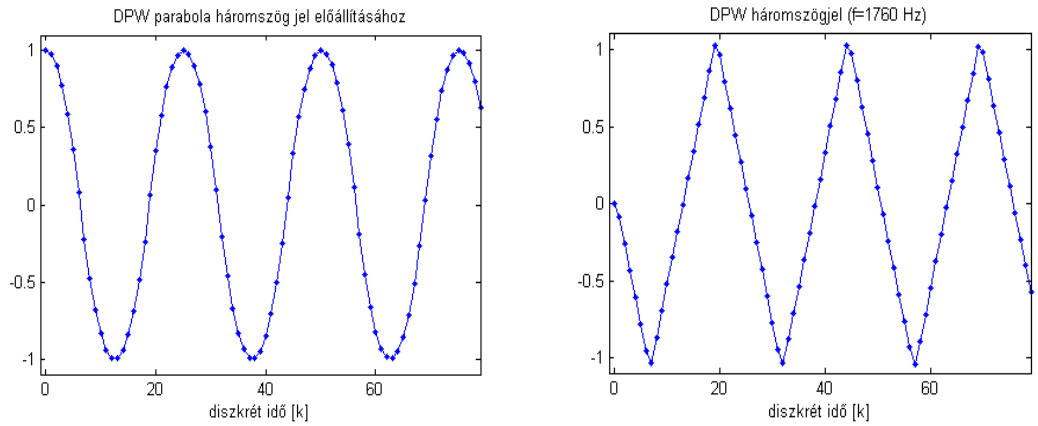
.. ábra Az ideális (balra), valamint a PolyBLEP (jobbra) négyszögjel spektruma (f=1760 hz)

Háromszögjel előállítása

Az ideális háromszögjel harmonikusai $1/f^2$ szerint csökkennek a frekvencia növekedtével. A fűrészjel és a négyszögjel 6 dB/oktávos csökkenésével szemben ez 12 dB/oktáv sebességgel csillapodó harmonikusokat jelent. Mindez azt eredményezi, hogy a háromszög triviális módon való előállítása jóval kisebb aliasing hibát okoz, mint a másik két hullámformánál. Emiatt akár a triviális megoldás is kielégítő eredményt adhat. A triviális megoldást megvalósíthatjuk például egy számlálóval, mely megváltoztatja a számlálás irányát, ha 1-hez, vagy -1-hez ér.

Amennyiben a triviális megoldásénál jobb minőségre van szükség, a triviális megoldás pl. kétszeres túlmintavételezése hatékony megoldás.

A DPW algoritmus is alkalmazható háromszög hullámforma létrehozására. A fűrészjel generálásához hasonlóan egy szakaszosan parabolikus hullám differenciálásával jutunk a kívánt eredményre.



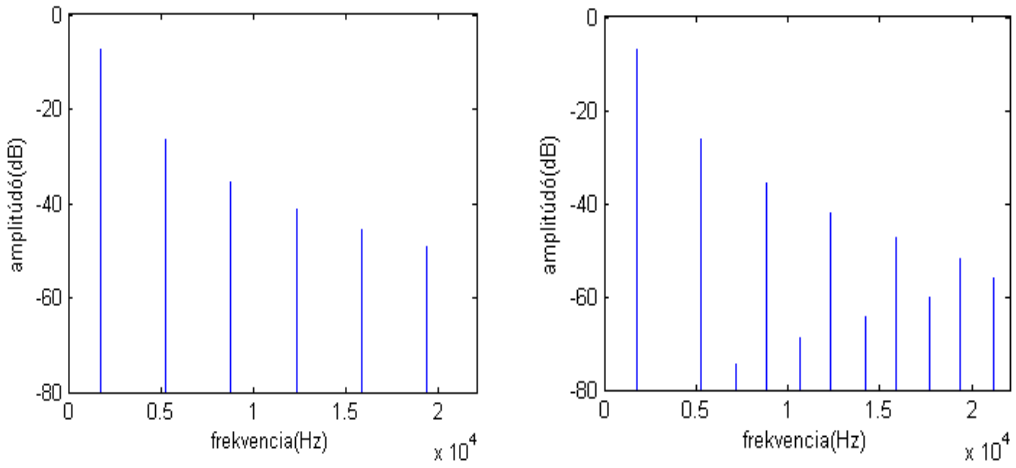
.. ábra Háromszögjel előállítása DPW algoritmussal (alapfrekvencia f=1760 Hz)

A 2.18-as ábra bal oldalán látható szakaszosan parabolikus hullámot a fűrészjelnél használt parabolát felhasználva kapjuk. A fűrészjelnél használt parabolát (2.13-as ábra) 1-ből ki kell

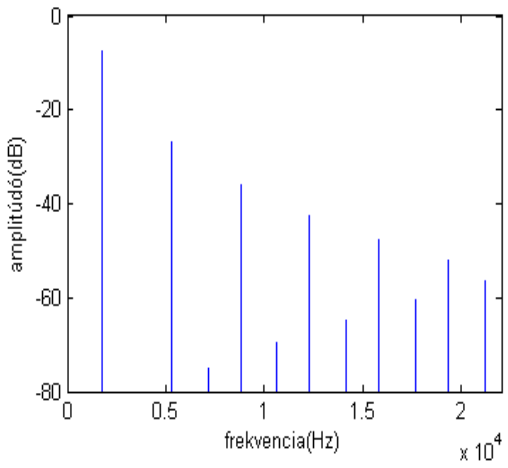
vonnunk, majd egy négyszögjellel megszoroznunk. A fűrészjel-féle parabolához használt számlálót kétszeres frekvencián kell futtatnunk a létrehozandó háromszög alapfrekvenciájához képest. A négyszögjel (amellyel beszorzunk) frekvenciája a háromszög alapfrekvenciájával megegyező kell, hogy legyen. A differenciálás és a skálázás a 2.11-es és 2.12-es képletekben megadott módon zajlik a háromszögjel esetében is.

Háromszögjelet állíthatunk elő négyszögjel integrálásával is, így például PolyBLEP módszerrel képzett négyszögjelből nagyon jó jel/zaj viszonnyal rendelkező háromszögjelet tudunk képezni.

A 2.19-es ábrán az ideális (additív szintézissel alkotott), valamint a DPW algoritmussal képzett háromszögjel spektrumát láthatjuk. A 2.20-as ábrán pedig egy PolyBLEP algoritmussal generált négyszögjel integrálásával létrehozott háromszögjel spektrumát ábrázoltam. A háromszögjel esetében a DPW és a PolyBLEP közti különbség szemmel szinte nem is látható.



.. ábra Az ideális (balra) és a DPW (jobbra) háromszögjel spektruma (f=1760 hz)



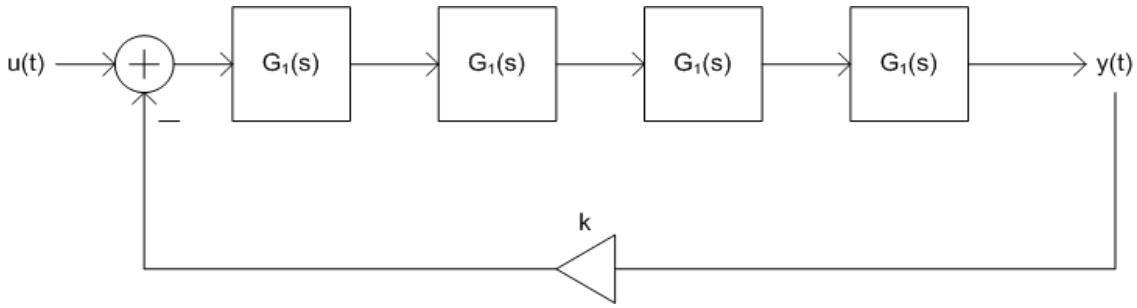
.. ábra PolyBLEP négyszögjel integrálásával képzett háromszögjel (f=1760 Hz)

3. A Moog szűrő digitális implementálása

Az oszcillátorok mellett a szubtraktív szintézist megvalósító szintetizátorok másik központi jelentőségű eleme a szűrő. Általában egy rezonáns aluláteresztő szűrőt szoktak alkalmazni, melynek segítségével az oszcillátorok nagy felharmonikus tartalmú jeleit formáljuk a kívánt hangzás elérésének érdekében. Szoftveres szintetizátoromban a klasszikusnak számító Moog analóg szintetizátor szűrőjét, a Moog szűrőt implementáltam. A fejezetben először bemutatom a Moog szűrőt (struktúráját, fontosabb jellemzőit, paramétereit), majd bemutatom digitális implementációjának általam alkalmazott módját.

3.1 A Moog szűrő

A Moog szűrő egy rezonáns aluláteresztő szűrő. Struktúrája az alábbi ábrán látható:



.. ábra A Moog szűrő felépítése

A Moog szűrő négy egyforma egypólusú aluláteresztő szűrőblokk kaszkádolásából áll, valamint rendelkezik egy negatív visszacsatolással. Az egypólusú szűrők átviteli függvénye a következő:

$$G_1(s) = \frac{1}{1 + \frac{s}{\omega_c}} \tag{3.1}$$

A 3.1-es képletben $s = j\omega$ a komplex frekvenciát, ω_c a vágási (kör-) frekvenciát jelöli. G_1 pólusa $s = -\omega_c$. A vágási frekvencián a következő értéket veszi fel:

$$G_1(j\omega_c) = \frac{1}{1+j} = \frac{1}{\sqrt{2}} e^{j\frac{\pi}{4}} \tag{3.2}$$

G_1 a vágási frekvencián 45 fokos fáziskésleltetést és kb. 3 dB-es amplitúdócsökkenést eredményez. A négy egymás után kötött egypólusú szűrő együttes hatását G_1 negyedik hatványra emelésével kapjuk meg. A négy szűrőből álló blokk átvitele a vágási frekvencián:

$$G_1^4(j\omega_c) = \frac{1}{4} e^{j\pi} = \frac{1}{4}(-1) \tag{3.3}$$

A Moog szűrő előre vezető ága (a felnyitott kör) a vágási frekvencián tehát körülbelül 12 dB-es amplitúdóesést és 180 fokos fázistolást okoz. Utóbbi azt jelenti, ezen a frekvencián a szűrő invertál. Az invertált jel a visszacsatolás (zárt kör) során mínusz eggyel szorzódik (negatív visszacsatolás), így a vágási frekvencián a Moog szűrő rezonanciát okoz.

A teljes Moog szűrő átviteli függvénye:

$$H(s) = \frac{G_1^4(s)}{1 + kG_1^4(s)} = \frac{1}{k + \left(1 + \frac{s}{\omega_c}\right)^4} \tag{3.4}$$

A 3.4-es képletet a szabályozástechnikából ismert, $\text{zárt kör átvitele} = \frac{\text{felnyitott kör átvitele}}{1+\text{felnyitott kör átvitele}}$ szabályból kaphatjuk meg. Az egyenletből látható, hogy a Moog szűrő pólusait k értéke befolyásolja. Amennyiben k 0 és 4 közé esik, a pólusok a negatív félsíkra esnek a komplex számsíkon, a szűrő stabil lesz. Ha k=0, nem lép fel rezonancia, a szűrő egy szimpla aluláteresztő viselkedést mutat. k=4 esetén pedig a rezonancia maximális, a szűrő ekkor oszcillál.

Az eredeti Moog szűrő rendelkezik néhány nagyon előnyös tulajdonsággal [6]:

- Vezérlése nagyon egyszerű. A szintetizátor használója közvetlenül állíthatja a vágási frekvencia, rezonancia paramétereket, ez könnyű kezelhetőséget biztosít az eszköznek.
- A vágási frekvencia és a rezonancia paraméterek egymástól függetlenek, az egyiken állítva nem tolódik el a másik értéke.
- A tranzisztoros megvalósítás következményeként nemlineáris viselkedést mutat a szűrő. Ez egyfajta karakteres hangzást ad az eszköznek.

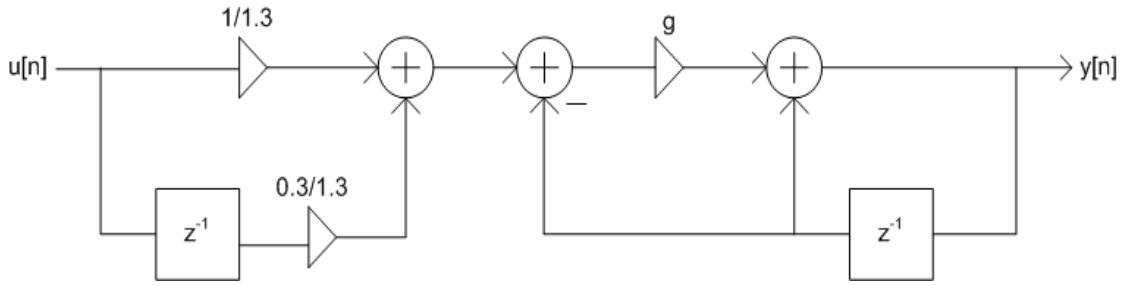
Az itt felsorolt tulajdonságokat szeretnénk megőrizni a digitális implementáció során is.

3.2 Digitális megvalósítás

Az analóg Moog szűrő digitális implementációjára számos lehetőség kínálkozik. Egy folytonos idejű rendszer diszkrét időbeli rendszerré konvertálása ugyanis nem triviális. Egy lehetőség, hogy valamilyen transzformációs módszert választunk, ami az s komplex frekvenciát leképezi a z diszkrét idejű komplex frekvenciába. Ilyen transzformációk például az impulzus-invariáns transzformáció, bilineáris transzformáció, az előretartó- és a hátratartó differencia módszerek. Másik lehetőség, hogy nem az analóg rendszert próbáljuk átkonvertálni, hanem közvetlenül digitális szűrőt tervezünk, amely a lehető legjobban közelíti az analóg rendszer viselkedését. A különböző módszerek mind-mind eltérő módon és mértékben őrzik meg az analóg szűrő fontos tulajdonságait.

Mindegyik módszernél problémát jelent a visszacsatolás megvalósítása. Diszkrét idejű rendszerben ugyanis ez csak minimum egy minta késleltetéssel realizálható (különben a kimeneti minta számításakor szükségünk lenne arra a mintára, amit éppen akkor számolunk ki). Ez a késleltetés azonban negatívan befolyásolja a szűrő tulajdonságait, többek közt megszűnik a rezonancia és a vágási frekvencia paraméterek egymástól való függetlensége [6].

Az általam megvalósított implementáció [7] az eredeti struktúrához hasonló. Szintén négy egypólusú szűrő sorba kötéséből, valamint negatív visszacsatolásból áll. Az egypólusú blokkok felépítése a következő:



.. ábra Az egypólusú szűrőblokk felépítése a Moog szűrő általam használt digitális implementációjában

A 3.2-es ábrán látható egypólusú szűrő átviteli függvénye:

$$H_1(z) = \frac{g}{1.3} \cdot \frac{z + 0.3}{z + (g - 1)} \tag{3.5}$$

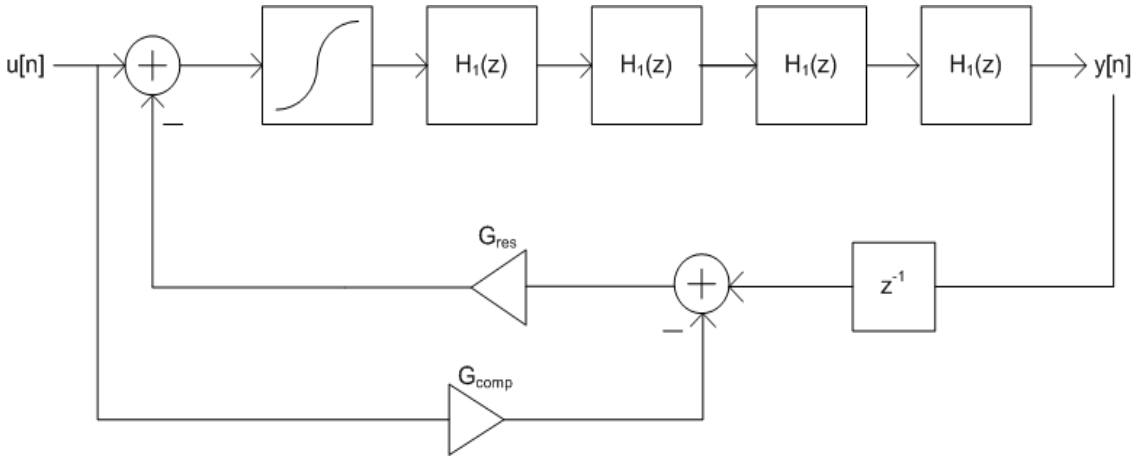
A 3.5-ös képletben látottak szerint ez a megoldás egy zérust (z=-0.3) illeszt a rendszerbe. Ennek segítségével nagyrészt kiküszöbölhető a vágási frekvencia és a rezonancia paraméterek egymástól függése, melyet az okoz, hogy a szűrő visszacsatoló ágába kénytelenek vagyunk egy késleltetést tenni. [6] A képletben látható g paramétert egy negyedfokú polinom adja:

$$g = 0.9892\omega_c - 0.4342\omega_c^2 + 0.1381\omega_c^3 - 0.0202\omega_c^4$$

$$\omega_c = \frac{2\pi f_c}{f_s} \tag{3.6}$$

ahol f_s a mintavételi frekvencia, f_c pedig a vágási frekvencia. A 3.6-os egyenlet polinomja „állítja be” az egypólusú szűrő pólusát oly módon, hogy a szűrő a megadott f_c vágási frekvenciának megfelelő működést produkálja.

A 3.3-as ábrán a digitális implementáció teljes struktúrája látható:



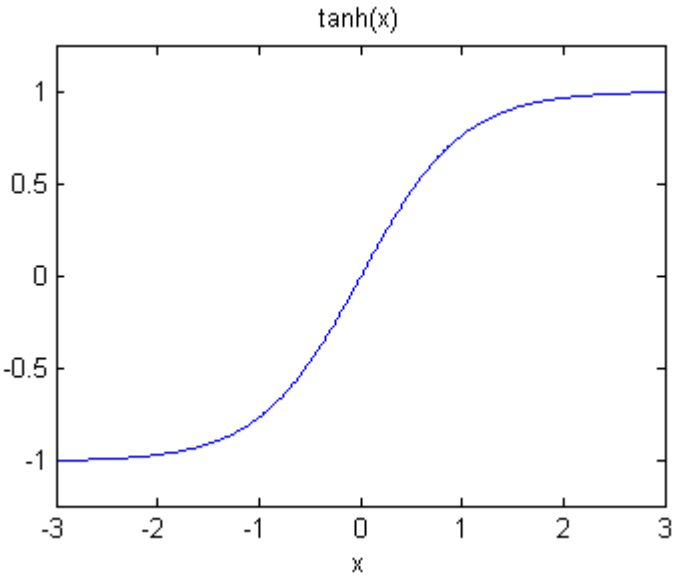
.. ábra A Moog szűrő digitális implementációjának struktúrája

Az eredeti analóg Moog szűrő felépítésétől némileg eltér ez a struktúra. A visszacsatoló ágban felfedezhető az említett, kényszerűségből beillesztett egy mintányi késleltetés. A G_{res} egy a g-hez hasonló funkciójú polinom (skálázási funkciót tölt be). Az analóg szűrő k paraméterének felel meg, a rezonanciát szabályozhatjuk vele. Arra szolgál, hogy a C_{res}-ben beállított ($0 \leq C_{res} \leq 1$) rezonancia paraméternek megfelelő rezonanciaműködést valósítsa meg a szűrő. G_{res} polinomja a 3.7-es képletben látható:

$$G_{res} = 4 \cdot C_{res}(1.0029 + 0.0526\omega_c - 0.0926\omega_c^2 + 0.0218\omega_c^3) \qquad (3.7)$$

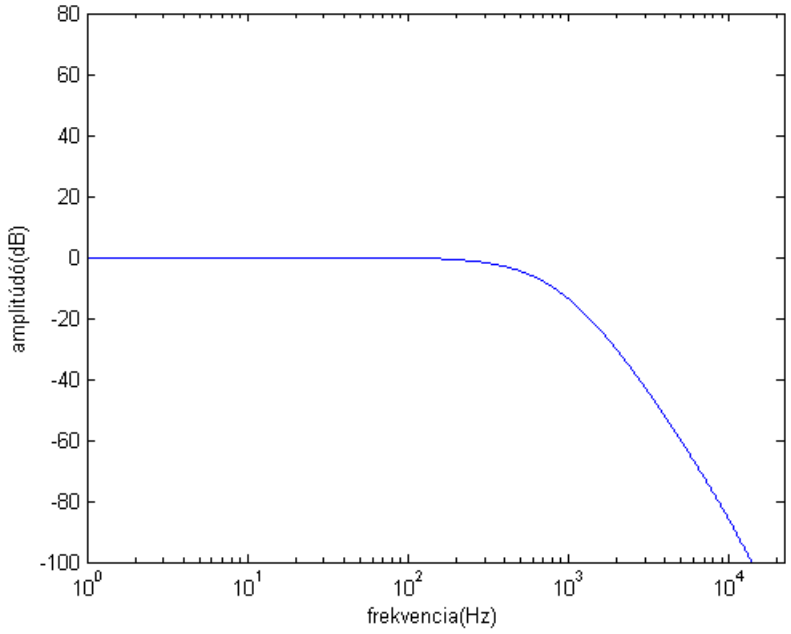
Egy másik eltérés az eredeti Moog szűrő felépítésétől, hogy a bemenő jelet G_{comp}-al súlyozott mértékben kivonjuk a visszacsatoló ágbeli jelből. Ezzel azt kompenzáljuk, hogy a rezonancia növelésével a szűrő áteresztő sávjának átvitele csökken. G_{comp} -ot 1-nek választva az áteresztő sáv átvitele konstans marad a rezonancia növelésével [7]. Ugyanakkor ez a kompenzáció a kimeneti amplitúdó nagymértékű növekedésével jár magas rezonancia értékek beállítása esetén. Emiatt G_{comp} értékét 0.5-re érdemes állítani. Hozzá kell tenni, hogy nem csak a digitális implementációra, de az eredeti Moog szűrőre is jellemző, hogy a rezonancia növelésével csökken az áteresztő sáv átvitele. Ezért ez a kompenzáció már nem az eredeti analóg szűrő szimulálásához tartozik, csak egy kiegészítő funkció melyet el is hagyhatunk, a szűrő működőképes és teljes mértékben használható nélküle.

Az utolsó új elem a blokkvázlatban egy nemlinearitás az egypólusú blokkok előtt. Ennek beiktatásával az eredeti analóg eszköz tranzisztoros felépítéséből adódó nemlinearitásokat próbáljuk szimulálni. A nemlineáris torzítás nem eredendő funkciója egy aluláteresztő szűrőnek, tulajdonképpen az ideális átviteltől való eltérés, tehát hibának tekinthető. Az eredeti Moog szűrőnek azonban ez a tökéletlenség egy egyedi hangzást kölcsönzött, melyet mi imitálni szeretnénk. Pontosabb eredményre jutnánk, ha minden egyes egypólusú szűrőblokkba nemlinearitást helyeznénk [8], azonban ez jelentősen növelné a számításigényt. Így a 3.3-as ábrán látható struktúra egy kompromisszumos megoldásnak tekinthető. A nemlinearitás konkrét megvalósítására használhatunk bármilyen folytonosan telítésbe menő függvényt. Implementációmban a tangens hiperbolikusz függvényt alkalmaztam.

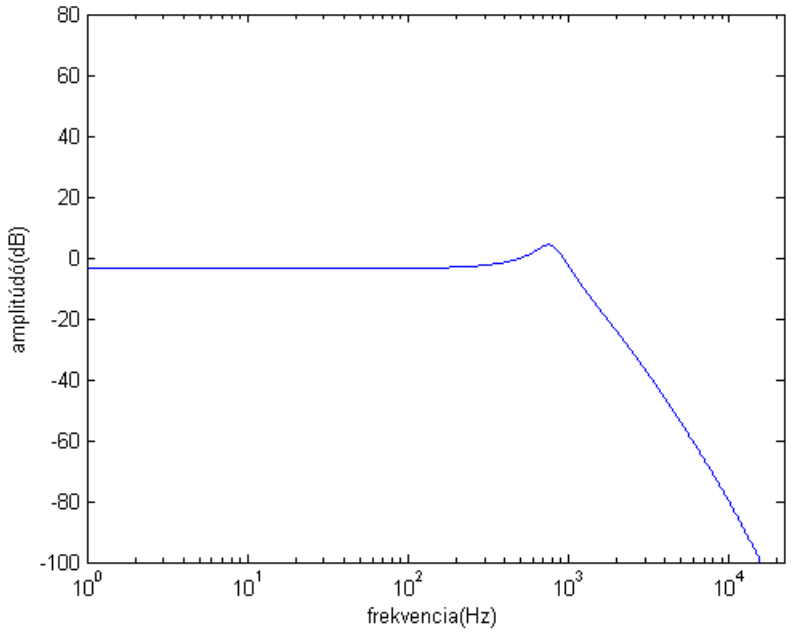


.. ábra A tangens hiperbolikusz függvény

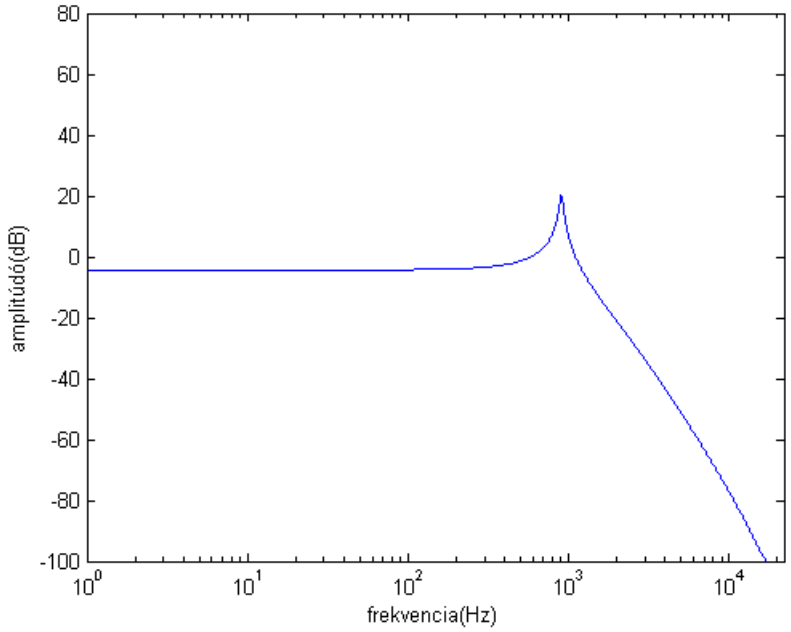
A 3.5-ös, 3.6-os, 3.7-es és 3.8-as ábrákon a megvalósított digitális Moog szűrő átviteli függvényét ábrázoltam adott vágási frekvencia (1 kHz), és különböző rezonancia (C_{res}) értékek esetén. Annak érdekében, hogy átviteli függvényt ábrázolhassak (csak lineáris rendszernek lehetséges átviteli függvényét megadni), az ábrák készítésekor kiiktattam a szűrőben lévő nemlinearitást. Ettől függetlenül az ábrák mérvadóak a szűrő átvitele szempontjából, mert a nemlinearitásnak csak nagy amplitúdók esetén van igazán jelentősége.



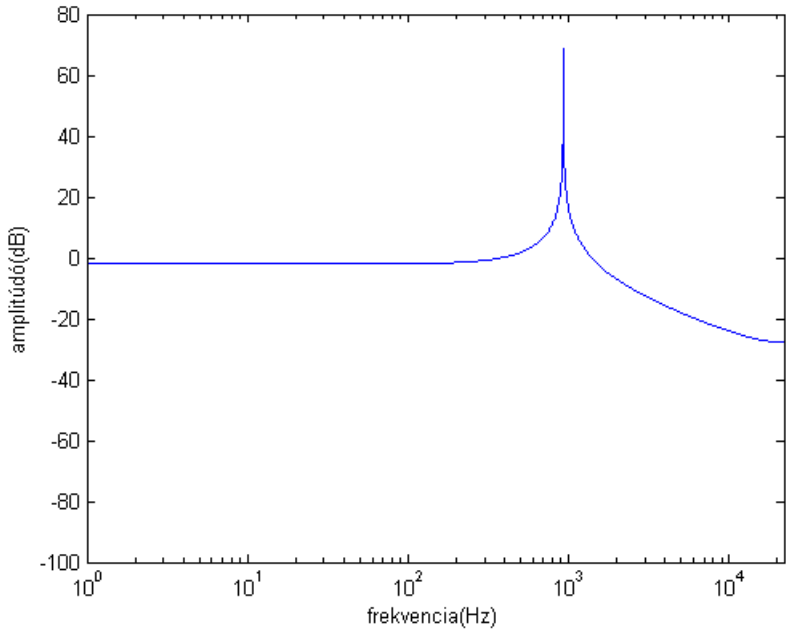
.. ábra A digitális Moog szűrő átvitele (fc=1 kHz, Cres=0)



.. ábra A digitális Moog szűrő átvitele (fc=1 kHz, Cres=0.5)



.. ábra A digitális Moog szűrő átvitele (fc=1 kHz, Cres=0.9)



.. ábra A digitális Moog szűrő átvitele (fc=1 kHz, Cres=1: a szűrő oszcillál, a vágási frekvenciánál egy szinuszos komponens meg a szűrt jelben)

Látható, hogy a rezonancia növekedtével kissé csökken az áteresztő sáv átvitele. A 3.4-es ábrán $C_{res} = 0$, nincs rezonancia. A 3.5-ös és 3.6-os ábrán látható, ahogy C_{res} növekedésével a vágási frekvencia környéki átvitel megnő, egyre nagyobb mértékben, és egyre szűkebb frekvenciasávban. A 3.7-es ábrán $C_{res} = 1$, a szűrő oszcillál. Ekkor a vágási frekvencián (a digitális implementációban, annak tökéletlensége miatt nem pontosan ott) egy szinuszos komponens jelenik meg a szűrt jelben.

4. Az implementációról

4.1 A VST környezet működéséről

Szoftveres szintetizátoromat a Steinberg cég Virtual Studio Technology (VST) fejlesztői környezetének [9] segítségével implementáltam. A VST olyan technológia, melynek felhasználásával audio effekteket, virtuális hangszereket hozhatunk létre. Az ingyenesen letölthető SDK (software development kit) segítségével létrehozott effektek, virtuális hangszerek úgynevezett plugin-ok. Futtatásukhoz egy gazdaprogram szükséges. Mivel a technológia olyannyira elterjedt, hogy már szinte szabványnak tekinthető, bőséggel találni gazdaprogramokat. Ezek a programok többnyire zeneszerkesztő programok, stúdióprogramok (pl. Cubase, Nuendo, Ableton), melyeket hangfelvételre és/vagy azok szerkesztésére, processzálására használhatunk.

A VST plugin-ok általában kétféle funkciót töltenek be. Egy részük olyan alkalmazás, melyet a gazdaprogram által kezelt audiosávok manipulálására használhatunk. Ezek audio effektek, pl. zengető, equaliser, stb. Másik részük virtuális hangszer, amit zenekészítéshez, felvételhez alkalmazhatunk, vagy mint szoftveres hangszert használhatjuk.

A VST plugin nem program, hanem függvények összessége. Windows platformon megjelenési formája egy függvénykönyvtár (DLL – Dynamic Link Library). A gazdaprogram a DLL-ben található függvényeket hívja meg a plugin működése során. A fejlesztés ezeknek a függvényeknek (illetve egy részüknek) a megírásából áll (C++ nyelven kell fejlesztenünk). Amennyiben audioeffektről van szó, a gazdaprogram az audio jelfolyamot adja át a plugin-nak (annak a függvénynek, amely a jelfeldolgozást végrehajtja,) ami visszatér a processzált jellel. Virtuális hangszer esetén a gazdaprogram a hangszer vezérlésének információit szolgáltatja a plugin-nak, ami ez alapján visszaadja a szintetizált jelet.

A gazdaprogram és a plugin közti kommunikáció blokkosítva zajlik. A gazdaprogram átadja az effekt, vagy virtuális hangszer bemenetéül szolgáló audio, vagy vezérlő információ egy blokkját, majd miután a plugin ezt feldolgozza, a kimenetet visszaadja. A gyakorlatban, pl. egy audioeffektnél a gazdaprogram egy memóriacímet ad át a processzáló függvénynek, mely a feldolgozandó audio jelfolyam egy blokkjának címe. A program átadja a blokk méretét is. A meghívott függvény, végrehajtván a blokk feldolgozását, egy szintén a gazdaprogram által megadott memóriacímre írja a művelet eredményét. Virtuális hangszer esetén ez annyiban más, hogy nincs audio bemenet, csak kimenet. A blokkosított kommunikációnak köszönhetően „valós időben” működhet a plugin.

A szoftveres szintetizátort VST virtuális hangszerként implementáltam. A virtuális hangszer vezérlése legtöbbször egy a számítógéphez kívülről kapcsolódó vezérlőeszköz, például billentyűzet segítségével történik. Leggyakrabban MIDI vezérlőt, pl. MIDI billentyűzetet használnak.

A MIDI egy aszinkron soros átviteli protokoll elektronikus zenei eszközök közötti kommunikációra [10]. A szintetizátorom esetében a számítógéphez kötött MIDI billentyűzeten[xv] lejátszott hangokról közvetít információt a számítógép felé.

A MIDI adatátviteli protokoll. A MIDI szabvány rengeteg féle üzenetet specifikál, melyek részletezésére nem térek ki. A szintetizátorom szempontjából lényeges üzenetek a NoteOn és a NoteOff üzenetek. A MIDI billentyűzet NoteOn üzenetet küld a számítógépnek, ha lenyomunk rajta egy billentyűt, NoteOff üzenetet, ha felengedjük. A MIDI üzenetek byte szerveződésűek [11]. Az üzenet első byte-ja a státusz byte, mely megmondja, milyen üzenetről van szó. NoteOn és NoteOff üzenet esetén két byte követi a státusz byte-ot. Az egyik byte a lejátszott hang magasságát hordozza (melyik billentyűt nyomtuk le/engedtük fel), a másik pedig a leütés/felengedés sebességét. Előbbi felel meg a lejátszott hang hangerejének. A leütés sebességéről szóló információt csak billentésérzékeny MIDI-billentyűzetek szolgáltatnak. Az implementált szintetizátorban csak a hangmagasság információt kezelem.

A VST gazdaprogram a lejátszott hangokról szóló MIDI információt (blokkokban) átadja a ProcessEvents nevezetű függvénynek. Egy-egy ilyen blokk pár ezredmásodpercnyi időt ölel fel. Az ez alatt az idő alatt bekövetkező MIDI eseményeket tartalmazza (mikor milyen billentyűt nyomtunk le/engedtünk fel). A hang szintetizálásához ezeket az információkat a ProcessEvents függvény dolgozza fel, majd a ProcessReplacing nevű függvény a MIDI események alapján szintetizált jelet a kimeneti bufferre írja. A plugin programozásának részleteire nem térek ki.

4.2 A megvalósított funkciókról

Ebben az alfejezetben röviden ismertetem a szoftveres szintetizátorban megvalósított funkciókat.

A szintetizátor két oszcillátort tartalmaz. Mindkettőnél választhatunk fűrész- és négyszögjel közül. A fűrészjelet és a négyszögjelet is a PolyBLEP algoritmus segítségével generálom. A két oszcillátor elhangolható negyedhanggal fölfelé, illetve lefelé, a hangolás mértékét centekben adhatjuk meg (egy cent félhangnyi távolság századrésze). A 3. fejezetben ismertetett rezonáns aluláteresztő szűrőt is beépítettem. Állítható paraméterei a vágási frekvencia, valamint a rezonancia (C_{res}). Előbbit 50 Hz és 20 kHz, utóbbit 0 és 1 között állíthatjuk. A szűrő nemlinearitása elé egy erősítőt tettem (egy egyszerű szorzással megvalósítva), amivel torzíthatjuk a jelet. A szintetizátor utolsó megvalósított blokkja egy kisfrekvenciás oszcillátor (LFO), melyet egy szinuszos oszcillátorként valósítottam meg. Frekvenciája 0 és 50 Hz között állítható. Az LFO-val három paramétert vezérelhetünk: hangerő, oszcillátor frekvencia, és a szűrő vágási frekvenciája. Az LFO-val kapcsolatos érdekes kérdés, hogy annak érdekében, hogy az LFO hullámformája, és az általa keltett változásérzet egybevágjon, valamilyen skálázás lehet szükséges. Tehát például ahhoz, hogy az amplitudót modulálva a hangerőváltozást szinuszosnak érzékeljük, át kell skáláznunk az LFO hullámformáját logaritmikusra, hogy ne közvetlenül az amplitúdó, hanem maga a hangerősség változzon szinuszosan. Ugyanez a gondolatmenet a frekvencia modulációjánál is felmerül. A zenei hangok minél magasabbak, annál távolabb

[xv] A ma kapható digitális szintetizátorok támogatják a MIDI protokollt, képesek MIDI billentyűzetként viselkedni.

vannak egymástól frekvenciában. Így ha közvetlenül a frekvenciát moduláljuk szinuszosan, akkor a hangmagasságérzet változása nem szinuszos lesz. Úgy fogjuk érezni, hogy a modulálás lefelé sokkal mélyebb hangokig viszi a hangmagasságot, mint felfelé. Mindezek miatt a modulált paramétereket nem közvetlenül az LFO értékével, hanem azt logaritmikusan skálázva modulálom. Az amplitúdó esetén ez a logaritmus 10-es alapú, a decibel skálának megfelelően. A vágási frekvencia és frekvencia modulációnál kettesalapú a logaritmikus skálázás, azt figyelembe véve, hogy a zenei hangközök frekvenciában arányt jelentenek, s ez az arány 2 valamely hatványa. Például fél hang távolság esetén a két hang frekvenciájának aránya $\sqrt[12]{2}$, míg oktáv hangköz esetén (egy oktáv 12 fél hangból áll) a két frekvencia aránya 2. Ez a megoldás sajnos nem tökéletes. A cél az, hogy egy frekvenciamodulált hang magasságérzete ne változzon a moduláció mélységével, tehát pl. egy „C” hangot modulálva mindvégig „C” hangot halljunk, csak különböző erősségű és sebességű vibrato-val. Ez azonban a fenti megoldást alkalmazva nincs így. Lehetséges, hogy más LFO hullámforma (pl. háromszögjel - lineáris), vagy skálázás szükséges. Az LFO praktikus megvalósításához tehát ez a téma további megfontolást igényel.

A paraméterek állításánál próbáltam a felhasználó számára kényelmes, praktikus állítási lehetőségeket biztosítani. Ennek keretében a hangerőket (oszcillátorok hangereje külön-külön, valamint a master hangerő) logaritmikus, decibel skálán állíthatjuk. A szűrő vágási frekvenciáját szintén logaritmikus skálán változtathatjuk (kettesalapú), így az alacsonyabb frekvenciákon pontosabb beállítás lehetséges.

.. ábra A VST plugin formájában implementált szoftveres szintetizátor kezelőfelülete a Cubase programban futtatva

Összefoglalás és továbbfejlesztési lehetőségek

Szakdolgozatomban az analóg szintetizátor digitális implementálásának kérdéseit vizsgáltam. Az első fejezetben rövid történeti áttekintés után az analóg szintetizátorok általános felépítését ismertettem. A dolgozat céljainak megfogalmazása után a második fejezetben a szakirodalom alapján részletesen elemeztem az aliasing problémát kiküszöbölő oszcillátor algoritmusokat, melyek közül a legígéretesebbeket (Additív, DPW, DPW2X, PolyBLEP) Matlab-ban implementáltam. Először az ideális megoldásról, az additív szintézisről írtam, mely tökéletesen sávkorlátozott jelet hoz létre, de számítási igénye nagy. Következőnek a közelítőleg sávkorlátozott módszereket vizsgáltam, melyek egy aluláteresztővel szűrt folytonos idejű jel mintavételezett verzióját állítják elő. A sort a BLIT algoritmussal kezdtem, amely sávkorlátozott impulzussorozat létrehozását teszi lehetővé. Ebből az impulzussorozatból klasszikus hullámformák (pl. fűrészjel, négyszögjel) sávkorlátozott verziói állíthatók elő. A BLEP módszer a jel diszkontinuitásait a BLEP residual hozzáadásával teszi sávkorlátozottá (bizonyos mértékig). A következő vizsgált algoritmus a BLEP elvét követő, de jóval egyszerűbb és kisebb számításigényű PolyBLEP volt. E módszer a diszkontinuitások előtti és utáni egy-egy minta módosításával igen jó eredményt produkál. Végül a spektrum esését módosító algoritmusokkal, a Lane-nel, és a DPW-vel foglalkoztam. Mindkét módszer a kívánt jel integráltját (vagy ahhoz hasonló jelet) állítja elő, aminek spektruma meredekebben esik a frekvencia növekedtével, így az átlapolódás mértéke kisebb. A kívánt jelet (az eredeti spektrum meredekséget) a mintavételezés utáni differenciálással állítják vissza. A vizsgált megoldásokat egy egyszerű kritériumrendszer segítségével hasonlítottam össze, ami alapján a PolyBLEP módszert választottam. A harmadik fejezetben a szubtraktív szintézis során alkalmazandó szűrő digitális megvalósításának egy lehetséges megoldását ismertettem. Ez a megoldás a klasszikus Moog szintetizátor rezonáns alul áteresztő szűrőjén alapul. A választott oszcillátor algoritmus és szűrő segítségével VST környezetben sikeresen implementáltam egy működő, MIDI-billentyűzettel vezérelhető virtuális analóg szintetizátort. Az implementációról a negyedik fejezetben írtam röviden.

A megvalósított szintetizátort többféle irányba is lehetséges továbbfejleszteni. A 2009 szeptemberében lezajlott 12. nemzetközi DAFx (Digital Audio Effects) konferencián újabb oszcillátor algoritmusok kifejlesztéséről számoltak be [12], melyekről 2010-ben várható publikáció. Az újabb módszerek vizsgálatával lehetséges, hogy az eddigieknél jobb megoldáshoz juthatunk. Egy másik fejlesztési irány a szűrő implementációjának részletesebb, teljes körű vizsgálata. A szintetizátor használhatóságán is érdemes lenne javítani. A megvalósított LFO funkció nem tökéletes, további kalibrálás szükséges például a frekvenciamoduláció kívánt működéséhez. A szintetizátor új funkciókkal való bővítése is kívánatos. Ilyen bővítés lehet az oszcillátorok választható hullámformái közé a háromszögjel beépítése, vagy az LFO modulációs módjainak kiegészítése a pulzusszélesség modulációval (PWM) négyszögjelek esetén. Egy ADSR modul megvalósítása is nagyban növelné a felhasználó variációs lehetőségeit, a szintetizátor használhatóságát.

Függelék

A CD melléklet tartalma

A CD melléklet négy mappát tartalmaz:

M file-ok: ebben a mappában a Matlab-ban kipróbált algoritmusokhoz írt függvények .m fájljai találhatóak meg. A Matlab-ban ezt a mappát workspace-ként beállítva meg tudjuk hívni ezeket a függvényeket. A függvények argumentuma a kívánt alapfrekvencia. Egy példa: y=saw_dpw(440). Ekkor az y vektorba egy 1 másodperc (44100 elem) hosszú DPW algoritmussal generált fűrészjel kerül. Az .m fájlok között található még a moog szűrő implementációja, valamint az SNR számításához, kirajzolásához használt függvények.

SawOsc: ez a mappa egy általam írt demo programot tartalmaz, mellyel a triviális, az additív, a DPW, a DPW2X és a PolyBLEP algoritmussal képzett három különböző frekvenciájú hangot hallgathatunk meg, valamint megtekinthetjük spektrumukat. A program futtatásához olyan Windows operációs rendszer szükséges, melyre telepítve van a Microsoft .NET keretrendszerének legalább 2.0-ás verziója.

VST-Host: a mappa tartalma egy ingyenesen letölthető VST plugin gazdaprogram, melyben kipróbálható az általam megvalósított virtuális analóg szintetizátor. A vsthost.exe elindítása után be kell töltenünk a plugint. A Ctrl+N billentyűkombinációval hívható elő az ehhez szükséges párbeszédablak. A plugin a VST-Host mappa PolyBlepSynth almappájában található vstPolyBlepSynth.dll fájl. Betöltés után a PlugIn/Window/Parameters paranccsal hozhatjuk elő a szinti paramétereit. Ha csatlakoztattunk MIDI billentyűzetet, akkor már ki is tudjuk próbálni. MIDI billentyűzet nélkül a View/Keyboard Bar paranccsal hozható elő egy billentyűzet a képernyő alján, melyen aztán az egérrel tudunk játszani (a bal gomb lenyomásakor addig szól a hang, míg nyomva tartjuk, jobb gombbal kattintva addig, amíg újra rá nem kattintunk az egér bal gombjával).

Forraskod: Ebben a mappában a plugin forráskódjának három fájlja található. Önmagában e három fájlból nem tudunk plugin-t fordítani. A VST SDK (2.4-es verzió) példa solution-jének azonos nevű fájlai helyébe másolva, a vstxsynth projektet fordítva kapjuk meg a plugin-t alkotó dll-t.

Irodalomjegyzék

[1] **Valimaki, Vesa és Huovilainen, Antti** 2007 March, *Antialiasing Oscillators in Subtractive Synthesis,* IEEE Signal Processing Magazine, old.: 116-125.

[2] **Valimaki, Vesa** 2005 March, *Discrete-Time Synthesis of the Sawtooth Waveform With Reduced Aliasing,* IEEE Signal Processing Letters, Vol.12, No.3., old.: 214-217.

[3] **Stilson, Tim és Smith, Julius** 1996, *Alias-Free Digital Synthesis of Classic Analog Waveforms*

[4] **[Online]** *Red Book (audio CD standard)*
http://en.wikipedia.org/wiki/Red_Book_(audio_CD_standard)

[5] **[Online]** *Note names, MIDI numbers and frequencies*
http://www.phys.unsw.edu.au/jw/notes.html

[6] **Stilson, Tim és Smith, Julius** 1996, *Analyzing the Moog VCF with Considerations for Digital Implementation*

[7] **Valimaki, Vesa és Huovilainen, Antti** 2006 Summer, *Oscillator and Filter Algorithms for Virtual Analog Synthesis*, Computer Music Journal, old.: 19-31.

[8] **Huovilainen, Antti** 2004, *Non-linear digital implementation of the Moog ladder filter*

[9] **Software Development Kit 2.0.** 1999, *Steinberg Virtual Studio Technology Plug-In Specification*

[10] **[Online]** *MIDI Basics Introduction into MIDI* http://www.midiworld.com/basics/

[11] **[Online]** *MIDI Messages* http://www.midi.org/techspecs/midimessages.php

[12] **Smith, Julius** 2009, *Recent CCRMA research in digital audio synthesis, proccessing, and effects*

[13] **Brandt, Eli** 2001, *Hard Sync Without Aliasing*

[14] **[Online]** *Analog synthesizer* http://en.wikipedia.org/wiki/Analog_synthesizer

[15] **[Online]** *Lebegés (hangtan)* http://hu.wikipedia.org/wiki/Lebegés_(hangtan)

[16] **Bristow-Johnson, Robert** 1996, *Wavetable Synthesis 101, A Fundamental Perspective*

[17] **[Online]** ***Doering, Ed*** *Analog Synthesis Modules* http://cnx.org/content/m15442/latest/

[18] **[Online]** *Synthesizer* http://en.wikipedia.org/wiki/Synthesizer