Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Analóg szintetizátor modellezése különböző oszcillátor algoritmusokkal

Szakdolgozat

Készítette
Ambrits Dániel

Konzulens
Dr. Bank Balázs

2012. december 10.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Ambrits Dániel*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelmûen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerzõ(k), cím, angol és magyar nyelvû tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhetõ elektronikus formában, a munka teljes szövegét pedig az egyetem belsõ hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetõvé.

Budapest, 2012. december 10.

Ambrits Dániel
hallgató

Kivonat

Az analóg szintetizátorok 60-as, 70-es évekbeli elterjedése óta töretlen népszerűségnek örvendenek. Azonban a gyártók ma már nem gyártják ezeket a modelleket, ezáltal igény van klasszikus hangzások emulálására. A hangszerek számítógépes szoftverként való megvalósításai alacsony áron érhetőek el. Az egyik legelterjedtebb ilyen szoftveres formátum a Virtual Studio Technology, azaz VST.

Az szakdolgozatom során egy ilyen analóg szintetizátort készítettem VST környezetben. A hangszer a szubtraktív szintézis, azaz széles spektrumú jel szűrésének elvén alapul. Megismerkedtem a szintetizátor belső egységeivel (a jelgeneráló oszcillátorral, a szűrővel, az előerősítővel és a moduláló oszcillátorral) és ezek digitális implementációival.

Megvizsgáltam a jelgenerálás triviális megoldásának lehetőségét, melynél megjelenik az átlapolódás jelensége. A problémát redukáló algoritmusok közül a Differentiated Parabolic Waveform és Polynomial Transition Regions módszereket vizsgáltam, melyek hullámformák polinomfüggvényeinek deriválásán alapulnak. Alkalmazásukkal az átlapolódás mértéke csökkent.

Mindkét algoritmust kiterjesztettem változtatható kitöltési tényezőjű háromszög előállítására, a pulzusszélesség modulálásának érdekében. A módszer alapján meghatároztam a jelgenerálás egyes lépéseit. A DPW a kitöltési tényező hirtelen változására érzékeny módszernek bizonyult, míg a PTR korlátozások nélkül használható. A további összehasonlítás során a PTR számításigénye kisebbnek bizonyult, így a szintetizátorban ezt az algoritmust implementáltam a szoftverben.

A szűrőt a Minimoog szintetizátorban található típus digitális modellje alapján valósítottam meg. Az implementáció kedvező tulajdonsága, hogy a megvalósított szűrő törésponti frekvenciájának modulálása során a tranziensek nem jelentősek. Az erősítőt egy ADSR típusú burkológörbe-generátorral vezéreltem.

A MATLAB-szimulációk után a szintetizátort átültettem VST környezetbe a Steinberg VST SDK csomagja segítségével C++ nyelven.

Abstract

Analog synthesizers have been constantly popular since their spread in the 60s and 70s. However, the manufacturers no longer produce these models, thus there is a need to emulate classic sounds and implement these musical instruments as a cheap computer software. Many of the most popular types of such software use the Virtual Studio Technology, or VST.

In the course of my thesis such an analog synthesizer has been made in VST environment. The instrument is based on subtractive synthesis, which consists in the filtering of a signal having a wide spectrum. The thesis discusses the internal units (the signal generator oscillator, the filter, the pre-amplifier, and the oscillator for modulation) and their digital implementations.

The trivial solution of the signal generation leads to aliasing. There are various algorithms which reduce the problem. In the thesis the Differentiated Parabolic Waveform and Polynomial Transition Regions are studied. These methods are based on differentiating the polynomial functions of the waveform, leading to reduced aliasing and better sound quality.

Both algorithms have been extended to produce triangle signals with variable duty cycle. Each step of the signal-generation has been defined on the basis of the original methods. DPW turned out to be sensitive to the sudden change of the duty cycle, while PTR can be used without restriction. Further comparsion pointed out that PTR is more cost-effective, hence this algorithm was implemented in the software synthesizer.

The digital filter is based on the classic Moog filter. Modulating the cutoff frequency of the implemented filter leads to negligible transients. The amplifier is controlled by an ADSR envelope generator.

Following the simulations in MATLAB, the synthesizer has been implemented as VST, written in C++ language with the help of the Steinberg VST SDK.

1. fejezet

Bevezetés

1.1. Történeti áttekintés

A 20. század zenei történetében nagy szerepet játszik a szintetizátorok, azaz mesterséges hangok előállítására alkalmas elektronikus eszközök fejlődése. A kezdeti modellek (pl. Theremin) után a szintetizátorok egyre nagyobb tudással rendelkeztek, nagyobb szabadságot adva a hangkeltésben. Sokáig ezeket az eszközöket hangszernek aligha lehetett nevezni, hiszen bonyolult kezelésük inkább programozói feladat volt, mint zenei, ráadásul hatalmas méretűek is voltak. Az áttörést a Robert Moog által fejlesztett Minimoognak tulajdonítják, mely 1970-ben került kereskedelmi forgalomba. Ez volt az első szintetizátor, mely hordozható volt, kezelése és izgalmas hangok előállítása egyszerűbb volt, vagyis egy kompakt hangszerként funkcionált. A következő évtizedekben az analóg szintetizátorok óriási népszerűségnek örvendtek, a 70-es, 80-as években egy a hangszerről elnevezett zenei stílus, a szintipop is a szintetizátorokra épült. Időközben megjelentek a szintetizátorok digitális változatai, a 90-es években pedig a PC-s szoftverek és pluginek formájában.

Több klasszikus analóg szintetizátor, mint pl. a Minimoog, Korg MS-20, Korg Polysix, Roland Juno-60, még mindig népszerűek, sőt, az utóbbi években a klasszikus szintetizátorok a reneszánszukat élik. Azonban ma már ezeket a hangszereket nem gyártják, illetve egyéb analóg szintetizátor is alig található a kínálatban. A régi szintetizátorok lassan elromlanak, javítani nehéz vagy egyáltalán nem lehet, ráadásul egy-egy használt darab elég drágán szerezhető be, ezáltal igény van olcsóbb megoldásokra, melyekkel hasonló hangzások állíthatóak elő. Kaphatóak ún. virtuális analóg szintetizátorok, melyek külsője és kezelése hasonlít a kompakt analóg szintetizátorokéra, a hangot pedig valójában egy processzor állítja elő. A kisebb modellek viszonylag jó áron kaphatóak, néhányuk, pl. a Microkorg igen nagy népszerűségnek örvendenek. Alternatív megoldás teljesen szoftveresen, pluginként szimulálni egy klasszikus hangzást. Egyes gyártók, mint a Korg, saját maguk adták ki klasszi-

1.1. ábra. A Minimoog és szoftveres verziója [1, 2]

kus modelljeik szoftveres verzióját, természetesen az eredeti hardvernél jelentősen alacsonyabb áron, és a hangminőségre sem lehet semmi panasz. A szoftverszintetizátorok közül számos ingyenesen elérhető.

A feladatom egy virtuális analóg szintetizátor szoftveres verziójának elkészítése, mely hangzásában és működésében a Minimooghoz hasonlít, és rendelkezik a szokásos paraméterezhetőséggel. A specifikáció előtt tekintsük át a modellezni kívánt szintetizátor működését.

1.2. A szubtraktív szintetizátor

A szintetizátorok hangképzés szempontjából több típusba sorolhatóak. Beszélhetünk additív szintézisről, mely során a különböző harmonikusok összegzésével alakul ki a hangzás, FM alapú szintézisről, hangminta alapúról, a teljesség igénye nélkül [3].

Az analóg szintetizátorok túlnyomóan a szubtraktív szintézis alapján működtek, a módszer viszonylagos egyszerűsége miatt. A hangkeltés egy harmonikusokban gazdag hullámforma szűrésén alapul. Egy ilyen hangszer már néhány elemből, egy oszcillátorból és egy szűrőből összeállítható. Az alap blokkvázlat az 1.2. ábrán látható.

Az egyes egységek funkciói:

Voltage-controlled oscillator (VCO): A feszültségvezérelt oszcillátor állítja elő az alap, nyers jelet. A vezérlést a leütött billentyű alapján kapja, meghatá-

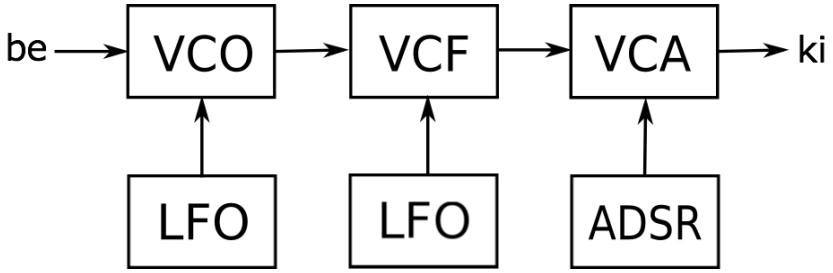
1.2. ábra. A szubtraktív szintetizátor váza

rozva a hang frekvenciáját. (Ez a frekvencia lesz a kimeneten kijutó hang frekvenciája is.) Többféle hullámformát lehet választani, a legáltalánosabbak a fűrészjel, a szimmetrikus háromszögjel, és szimmetrikus négyszögjel.

Voltage-controlled filter (VCF): A generált harmonikusokban gazdag jelet egy feszültségvezérelt szűrő szűri. Legtöbbször aluláteresztő, de találkozhatunk felül- és sáváteresztő modellekkel is. Az általánosan állítható paraméterei a töréspontí frekvencia és a rezonancia mértéke. Ez a szubtraktív szintézis legfontosabb eleme, ez határozza meg leginkább a hangzást.

Voltage-controlled amplifier (VCA): Egy előerősítő, mely felerősíti a hangot, mielőtt a kimenethez érne. Az erősítésének szabályozásával lehet a hangerőt állítani.

Ez mindössze a szintetizátor váza, a hangszerekbe további kiegészítő egységeket szoktak építeni, melyek több lehetőséget és sokkal nagyobb szabadságot nyújtanak egy hangzás előállítása során. A szakdolgozatban megvalósított szintetizátor blokkvázlata a 1.3. ábrán látható.



1.3. ábra. A megvalósított szintetizátor blokkvázlata

A kiegészítő egységek funkciói:

Low frequency oscillator (LFO): Általában 20 Hz-nél kisebb, alacsony frekvenciájú hullámformát előállító oszcillátor. A alapelemek, jelen esetünkben az oszcillátor és a szűrő különböző paramétereinek modulálására szokás használni, színesítve a hangzást. Jellemző hullámformái a szinuszjel, négyszögjel, háromszögjel, és a fűrészjel.

Attack Decay Sustain Release (ADSR): Az ADSR a burkológörbe-generátorok egy speciális fajtája, mellyel a hangerőt lehet szabályozni a billentyű leütésekor és elengedésekor.

1.3. Specifikáció

A működés ismeretében már specifikálni tudjuk, milyen követelményeknek feleljen meg a virtuális analóg szintetizátor.

- Szubtraktív szintézisen alapuló virtuális analóg szintetizátor
- VST plugin
- MIDI-kezelés
- VCO vezérelhető paraméterei:
  - hullámforma (fűrész, háromszög, négyszög)
  - frekvencia (hangmagasság)
  - hangerő
  - kitöltési tényező (modulálható LFO1 által)
  - LFO1 moduláció mélysége
- VCF a klasszikus Moog szűrőhöz hasonló
- VCF vezérelhető paraméterei:
  - törési frekvencia (modulálható LFO2 által)
  - rezonancia
  - LFO2 moduláció mélysége
- ADSR vezérelhető paraméterei:
  - attack, decay, sustain, release

A következőkben áttekintjük az egyes elemek digitális modelljét és implementációját MATLAB környezetben, majd VST pluginként való megvalósítását.

2. fejezet

VCO

Az oszcillátor két paraméter alapján állítja elő a kívánt digitális jelet: ezek a hang frekvenciája és a mintavételi frekvencia. Ezek alapján sorban generál mintákat a hullámformának megfelelő értékekkel. Hallható hangot az így mintavételezett jel folytonos idejűvé alakításával kapunk.

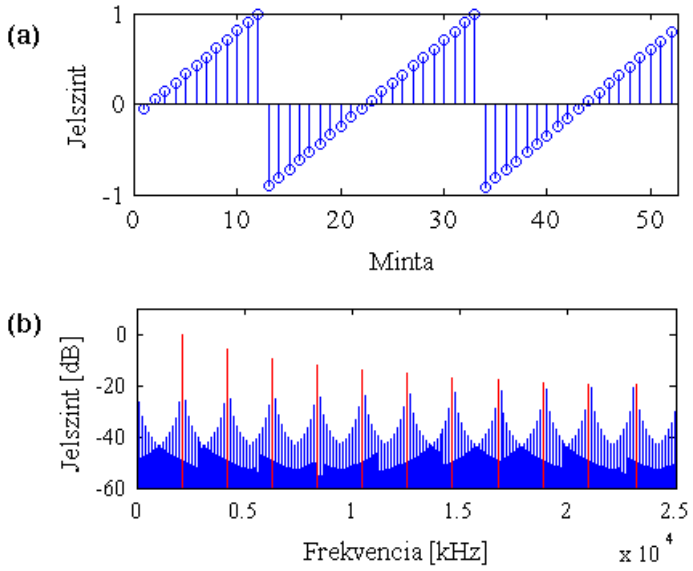
A továbbiakban tekintsük át a különböző hullámformák generálásának módjait. A mintavételi frekvenciát tekintsük állandónak, mely értéke 44,1 kHz, valamint az ábrákon a generált jelek frekvenciája 2093 Hz, ami a C_7 hangnak felel meg [4].

2.1. Triviális előállítás

A digitális jel triviális módon az analóg jel mintavételezésével állítható elő. A megvalósítás lényegében egy számláló, mely pl. fűrészjel esetében a hang frekvenciája által meghatározott meredekséggel számol -1-től 1-ig. Amikor a minta értéke 1-nél magasabb lenne, kivonunk belőle 2-t, ezáltal az érték annyival lesz magasabb -1-nél, mint amennyivel 1-nél lett volna magasabb [5, 6]. Így a jel a mintavételezett analóg fűrészjelnek felel meg. Az ebből adódó probléma a 2.1. ábrán látható.

A mintavételezés hatására a jel spektrumában az analóg spektrum a mintavételi frekvencia szerint periodikus lesz. Mivel az ugrások miatt a nagyfrekvenciás komponensek is jelentősek, a fűrészjel spektruma végtelen széles, ezért nem felel meg a Nyquist-kritériumnak. Ennek megfelelően érvényesül az átlapolódás hatása, azaz a spektrum különböző ismétlődéseiből származó komponensek összeadódnak, nem kívánt harmonikusok jelennek meg, amik, a jelet visszaalakítva analóggá és meghallgatva, zajszerű hatást okoznak a hangzásban. Az ideális fűrészjel spektruma a 2.1. ábrán pirossal van jelölve, jól látható az átlapolódásból adódó nagy mennyiségű nem kívánt harmonikus.

A cél tehát olyan hullámformák generálása, amelyek spektruma korlátozottabb, így a kisebb átlapolódás hatására a spektruma sokkal jobban közelíti az analóg jelét.



2.1. ábra. (a) Triviálisan generált fűrészjel és (b) spektruma

Erre számos algoritmust fejlesztettek ki. Az előző példához hasonlóan a továbbiakat is a fűrészjel generálásán keresztül mutatjuk be.

2.2. Korábbi algoritmusok

Additív szintézis

Az ideális spektrum eléréséhez legkézenfekvőbb az lenne, ha csak a szükséges harmonikusokat állítanánk elő a mintavételi frekvencia feléig, így az átlapolódás nem jelenne meg. Ez az additív szintézis. Mivel minden periodikus jel előáll szinuszok összegeként, ezért ehhez felhasználhatjuk a jel Fourier-sorát. Pl. a fűrészjel Fourier-sora:

$$s(n) = \frac{2}{\pi} \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\sin(2\pi k n f / f_s)}{k} \tag{2.1}$$

Ezt a diszkrét idejű jel szintézisekor nem végtelenig összegezzük, hanem csak $\frac{f_s}{2f}$ egészrészéig, így a legnagyobb frekvencián lévő komponens is a Nyquist-frekvencia alatt marad. Ezt az eljárást azonban nem érdemes használni, mivel főként alacsony frekvenciákon óriási a számításigénye. Minden egyes minta számolásához a harmonikusok számával egyenlő számú szinusz értéket kell kiszámítani és ezeket összegezni. Ennél jóval kisebb erőforrás felhasználásával is elő lehet állítani elfogadható minőségű jelet.

Az additív szintézis a teljesen sávkorlátozott lehetőségek közé tartozik. A következőkben nézzük át az úgynevezett kvázi-sávkorlátozott megoldásokat.

Bandlimited Impulse Trains

A kvázi-sávkorlátozott módszerek elve szerint egy aluláteresztővel szűrt folytonos idejű jel mintavételezett reprezentációját állítjuk elő, ezáltal közel átlapolódás-mentes hullámforma keletkezik.

A Bandlimited Impulse Trains (sávkorlátozott impulzussorozatok - BLIT) algoritmusában egy Dirac-impulzusokból álló

$$x(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT) \tag{2.2}$$

folytonos jelből szeretnénk a különböző hullámformákat előállítani. A fűrészjel az impulzussorozat integrálásával generálható:

$$s(t) = -\frac{1}{2} + \int_0^t \delta(\tau) - C_1(d\tau) \tag{2.3}$$

ahol $C_1 = \int_0^T \frac{\delta(\tau)}{T} d\tau$ DC offset, hogy a jel felvegye a fűrész alakot. Ahhoz, hogy a jel sávkorlátozott legyen, a felhasznált impulzussorozatot is sávkorlátozottá kell tenni. Ehhez aluláteresztővel meg kell szűrni, azaz egy alulátersztő impulzusválaszával vett konvolúciójára van szükség. Ez annak felel meg, hogy mindegyik impulzust kicseréljük az impulzusválaszra, és ezeket összegezzük [7].

Az ideális aluláteresztő a sinc függvény, így megoldásnak tűnik minden impulzus helyére egy-egy sinc függvényt másolni. Azonban a sinc függvény mindkét irányban végtelen, így ablakozásra van szükség a gyakorlati használatához. Az ablakozott sinc jel egy törtrészkésleltető szűrőnek felel meg. Ilyen szűrőkből léteznek más, pontosabb megvalósítások is, pl. B-spline vagy a Thiran mindentáteresztő szűrő, sinc helyett ezeket is szokás használni. A törtrészkésleltető angol neve, a fractional delay filter nyomán ezt az eljárást BLIT-FDF-nek is hívják [8]. Az ablak méretének növelésével pontosabb jelet lehet előállítani, azonban ezzel együtt nő a számításigény is.

Bandlimited Step Sequences

A valós időben történő integrálás viszonylag erőforrás-igényes, ezért érdemes elkerülni. A Bandlimited Step Sequences (BLEP) módszer erre nyújt megoldást, ami szerint a hullámformát egy növekvő lineáris jel és az úgynevezett BLEP maradék összegeként képezzük [7].

A módszer lényege a következő. A fűrészjel (és pl. négyszögjel) szakadásokat tartalmaznak. Ezek a végtelen meredekségnek felelnek meg, azaz az egységugrásnak, aminek spektruma végtelen. Ezt a spektrumot szeretnénk korlátozni. Az egységugrás a Dirac-impulzus integrálásával kapható. Ha a diszkrét idejű egységimpulzus helyett sávkorlátozott impulzust integrálunk, az ugrás is sávkorlátozott lesz. A BLEP

maradék nem más, mint sávkorlátozott és a triviális ugrás különbsége, így korrigálva az előállított jelet. A sávkorlátozott impulzus az (ablakozott) sinc függvény, ennek integrálását, és a különbségképzést pedig előre elvégezzük, így létrehozva a BLEP maradék táblázatot. A táblázatban lévő értékekkel kell a jelet korrigálni. Amire az alkalmazás során figyelni kell, hogy mivel a szakadáshoz tartozó korrekció a táblázat közepén helyezkedik el, előre figyelni kell az ugrás helyét, és eszerint végezni az összegzést.

Ez a nehézség megszüntethető minimálfázisú szűrő használatával. A minimálfázisú sinc impulzus aszimmetrikus, a szakadáshoz tartozó korrekció az első mintája, ezért nem kell előre megkeresni a szakadás helyét. Ez az eljárás a MinBLEP [9].

Polynomial Bandlimited Step Function

A Polynomial Bandlimited Step Function (PolyBLEP) a BLEP algoritmus egy továbbfejlesztése, ahol az integrálások és táblák helyett egy zárt alakban felírható polinomfüggvényt használunk a szakadás környezetének korrigálására [7].

A BLEP-hez hasonló módon egy aluláteresztő szűrő impulzusválaszára van szükség, amit majd integrálva sávkorlátozott ugrást kapunk. Az egyik legegyszerűbb függvény erre a célra egy háromszög:

$$s_{tri}(t) = \begin{cases} t+1 & \text{ha } -1 \leq t \leq 0 \\ 1-t & \text{ha } 0 < t \leq 1 \\ 0 & \text{egyébként} \end{cases} \tag{2.4}$$

Aminek a szakadással vett konvolúciója, azaz az integrálja:

$$s_{int}(t) = \begin{cases} 0 & \text{ha } t < -1 \\ \frac{t^2}{2} + t + \frac{1}{2} & \text{ha } -1 \leq t \leq 0 \\ t - \frac{t^2}{2} + \frac{1}{2} & \text{ha } 0 < t \leq 1 \\ 1 & \text{ha } 1 < t \end{cases} \tag{2.5}$$

ahol az $\frac{1}{2}$ konstanst a jelnek megfelelően választottuk. Ebből az egységugrást kivonva kapjuk meg a BLEP maradékhoz hasonló PolyBLEP függvényt:

$$p_{PolyBLEP}(t) = \begin{cases} \frac{t^2}{2} + t + \frac{1}{2} & \text{ha } -1 \leq t \leq 0 \\ t - \frac{t^2}{2} - \frac{1}{2} & \text{ha } 0 < t \leq 1 \end{cases} \tag{2.6}$$

A korrekció mindössze két, a szakadás előtti és utáni mintát módosítja. A BLEP ennél nagyobb korrigálást végzett, azaz a PolyBLEP (egyszerűsége miatt is) csak megközelíti a BLEP eredményeit, ellenben kisebb az erőforrásigénye. Jobb minőség eléréséhez a módszert ki lehet terjeszteni magasabb fokú polinomokra is, ezzel finomítva a korrekciót, a módosított minták száma pedig ezzel arányosan növekszik. Az

eredmény tovább javítható egy spline bázisfüggvényeket alkalmazó optimalizációs eljárással is [10].

2.3. Spektrum meredekségét növelő algoritmusok

Végül két olyan algoritmust tárgyalunk, melyek nem a Nyquist-kritériumnak megfelelően akarják korlátozni a spektrumot, hanem a harmonikusok csökkenésének sebességét növelik, így az átlapolódás ugyan megtörténik, de hatása jóval kevésbé érzékelhető, megfelelő hangminőséget produkálva. Az alábbi módszereket implementáltuk is, valamint saját eredményként alkalmassá váltak modulálható kitöltési tényezőjű háromszögjelek előállítására is.

2.3.1. Differentiated Parabolic Waveform

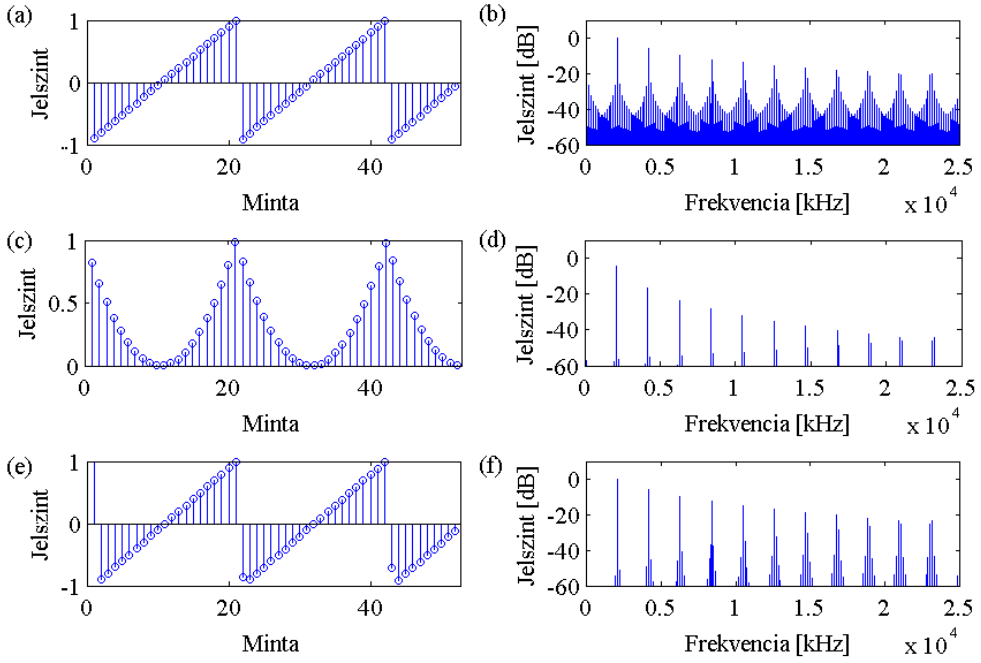
Az első ilyen módszer a Differentiated Parabolic Waveform (DPW) algoritmus [6]. Ebben az algoritmusban nem az előállítandó jelet mintavételezzük, hanem az integrálját. A szakasosan lineáris hullámformák, mint a fűrészjel vagy háromszögjel integrálja szakaszosan $x^2/2$, innen a "parabolikus" elnevezés. Az integrált jel spektruma kétszer akkora dB értékkel csökken oktávonként, mint a triviális jelé, így ekkor mintavételezve az átlapolódás sokkal kisebb mértékű. Ezután deriválással visszakapjuk a kívánt jelalakot.

A DPW-nek létezik továbbfejlesztett változata, mely a Differentiated Polynomial Waveform nevet viseli [11]. Ebben a triviális alapjelnek vesszük egy meghatározott N-edfokú polinomfüggvényét. Ennek a spektruma $6N$ dB értékkel csökken oktávonként, mintavételezéskor így az átlapolódott komponensek szintje jelentősen kisebb, és végül ezt a jelet $(N-1)$-szer deriváljuk. N-t növelve egyre kisebb az átlapolódás mértéke. A számításigény természetesen ennek megfelelően növekszik. Mivel a négyzetre emelés is polinomfüggvény, a Differentiated Parabolic Waveform az $N = 2$ esetnek felel meg. Nagyobb fokszámokra az elv hasonló, azokat részletesen a továbbiakban nem taglaljuk.

Fűrészjel

A szintetizátorokban található oszcillátorok egyik tipikus hullámformája a fűrészjel. A jel a 2.2. ábrán látható módon valósítható meg. A triviális fűrészjel négyzetre emelésével kapható meg a megfelelő parabola hullámforma, amit deriválva a generált jel időtartományban hasonlít a fűrészjelre, és az átlapolódás is sokkal kisebb mértékben jelenik meg.

Kirívó különbség látható a triviális és a DPW szerint előállított jelek spektrumában. A jobb hangminőség hallgatással is igazolható.



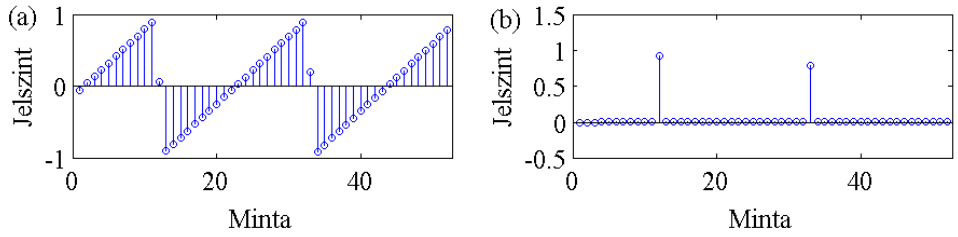
2.2. ábra. Fűrészjel generálása DPW algoritmussal: (a) triviális alapjel és (b) spektruma, (c) a négyzetre emelés utáni jel és (d) spektruma, (e) a deriválás és skálázás utáni végső jel és (f) spektruma

Az algoritmus MATLAB-os implementációjának egy módja az alábbiakban látható.

```
cntr = cntr + f / fs;
if cntr > 1.0
        cntr = cntr - 2;
end
sq = cntr^2;
dsq = sq - state;
state = sq;
out = c * dsq;
```

A kód egy mintát állít elő, és ciklusosan hívódik meg. A számláló értékét négyzetre emelve következik a deriválás. A deriválás az $1 - z^{-1}$ átviteli függvényű szűrőnek felel meg. A state változó az előző minta, ezt mindig felülíródik a jelenlegivel a következő minta számolásához. A differenciálást követően a jelet kompenzálni kell az alapfrekvencia függvényében, hogy a végső jel is -1 és 1 közötti értékeket vegyen fel. Ennek értéke $c = f_s/(4f(1 - f/f_s))$, ahol f_s a mintavételi frekvencia, és f a hang frekvenciája.

A 2.3 képen látható maga a generált hullámforma, és triviálishoz képesti különbségfüggvénye. Észrevehető, hogy a jel mindössze egyetlen mintában tér el a triviálistól, mégpedig a szakadás környékén. Ennek köszönhetően kevésbé meredek



2.3. ábra. (a) DPW módszerrel generált fűrészjel és (b) a triviális jeltől való különbség

a törés, így a magasabb frekvenciák amplitúdója jóval kisebb, így az átlapolódás is kevésbé észlelhető.

Háromszögjel

A fűrészjel után tekintsük át a szimmetrikus háromszögjel létrehozását. A háromszögjel spektruma 12 dB-lel csökken oktávonként, nem 6 dB-lel, mint a fűrész és négyszögjel esetében, az átlapolódás sokkal kisebb problémákat okoz, így akár a triviális módszer is megfelelő lenne, azonban törekszünk a még jobb hangminőségre.

A jel DPW algoritmussal történő előállítása hasonlít a fűrészjeléhez, ahogy az 2.4. ábrán is látható. A triviális háromszögjelet négyzetre emeljük, azonban itt ezután a jelet meg kell szorozni egy négyszögjellel. Ez a szorzás azért felel, hogy a jel megfelelő részei negatívak legyenek (itt fog a háromszögjel csökkenni), valamint folytonossá teszi a deriválandó jelet. Ezután következik a szokásos deriválás [5].

A triviális jel egy számláló -1 és $+1$ között, váltakozó számlálási iránnyal. Az új érték $+1$ fölé érve $2 - x$, -1 alá kerülve pedig $-2 - x$. Az algoritmusban használt négyszögjel a triviális jellel együtt generálható, az aktuális számlálási irány függvényében kell minden mintának 1 vagy -1 értéket adni. A két jel összeszorzását követően alakul ki a folytonos parabolikus jel, amit deriválva létrejön a kívánt háromszögjel.

Négyszögjel

A harmadik tipikus alapjel, a (szimmetrikus) négyszögjel előállítására több lehetőség is adódik.

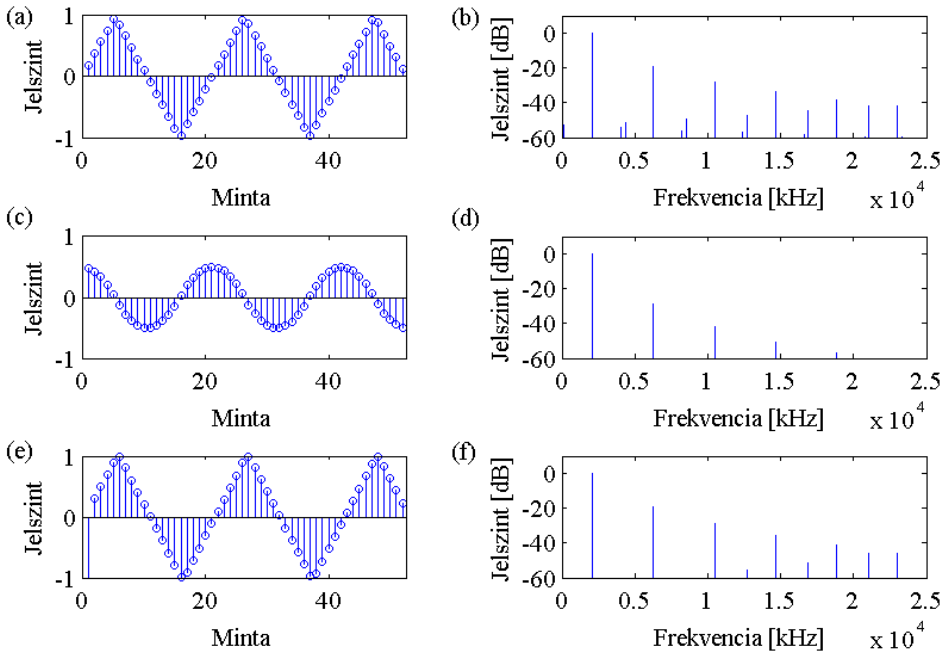
Az egyik, hogy két, egymáshoz képest megfelelően (fél periódussal) eltolt fűrészjel különbségét vesszük. Amennyiben triviális jelekkel végezzük el a generálást, a fűrészjel esetéhez hasonlóan nagy átlapolódást tapasztalhatunk. A jelenség kiküszöbölhető, ha DPW algoritmus által előállított fűrészjeleket használunk. Mivel ekkor a két jel különbségének abszolútértéke minden szakaszon 1 lesz, ezért további skálázásra nincs szükség. Ekkor a számításigény természetesen a fűrészjelének kétszerese.

A másik mód a háromszögjel deriválása. Mivel a két szakaszon a meredekség állandó, és csak előjelben különbözik, pont négyszögjelet kapunk eredményül. Hasonlóan az eddigiekhez, a deriválást a DPW algoritmussal generált háromszögjelen végezzük. Az eredményt skálázni kell, hogy -1 és $+1$ között mozogjon a jel, ehhez $c = f_s/4f(1 - f/f_s)$ értékkel kell megszorozni (hasonlóan a fűrészjelhez) [6].

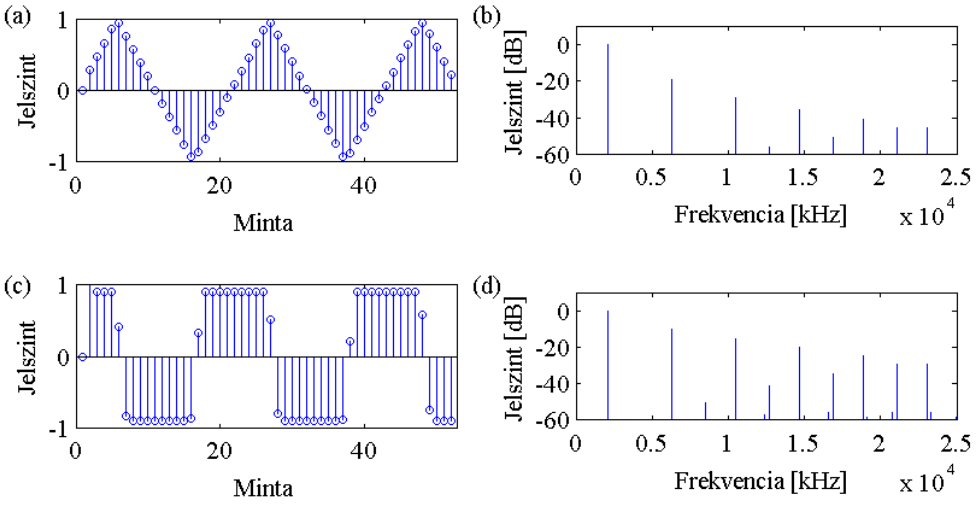
2.3.2. Polynomial Transition Regions

A DPW algoritmussal tehát viszonylag könnyedén előállítható a megfelelő spektrumú alapjel, azonban a számításigény még javítható, ugyanis minden egyes mintán elvégezni a négyzetre emelést, deriválást annak ellenére, hogy jórészt egyenletes meredekségű jeleket állítunk elő, feleslegesnek tűnhet. Erre nyújt megoldást a Polynomial Transition Regions módszer, mely ugyanazt a jelet állítja elő egyszerűbben és gyorsabban, mint a Differentiated Polynomial Waveform [12]. (A megközelítés hasonlít ahhoz, amit a BLEP és PolyBLEP esetében is láthattunk, azzal a különbséggel, hogy a PolyBLEP nem pontosan ugyanazt a jelet állította elő, mint a BLEP, csak megközelítette azt.)

Az elv azt a tulajdonságot használja ki, hogy az N-edfokú DPW algoritmussal előállított jel periódusonként mindössze N mintában (és fél mintavételi idővel való elcsúszásban) különbözik a triviális jeltől. Ezek a különböző minták a fűrész-, és



2.4. ábra. Háromszögjel generálása DPW algoritmussal: (a) triviális alapjel, (c) a parabolikus hullámforma,(e) a deriválás utáni végső jel, és az ezekhez tartozó spektrumok



2.5. ábra. (a) Háromszögjel és (b) spektruma, (c) a deriválással képzett négyszögjel és (d) spektruma

négyszögjel esetén a törések, háromszögjel esetén a csúcsok környezetében találhatók, a többi szakaszon esetleg csak kis offset eltérés található. A módszer használatához előre kiszámoljuk a DPW-vel generált hullámforma mintáit általánosan a töréspont környezetében (átmeneti régió) és ezen környezetén kívül. Az átmeneti régión kívül a jel a triviális jelnek felel meg, így a számláló növelésénél bonyolultabb számításokra nincs szükség.

A továbbiakban szintén csak az $N = 2$, azaz másodfokú esettel fogunk foglalkozni.

Fűrészjel

Tekintsük át a tipikus jelek generálásának módját ezzel a módszerrel is, kezdve a fűrészjellel. Ehhez vegyük át, mi történik a DPW generálás során.

Legyen egy p alapjelünk, mely egy f frekvenciájú fűrészjel (mod számláló) 0 és 1 között. Két szomszédos minta között az eltérés $f/f_s = T_0$. Ekkor $p(n) = nT_0 \bmod 1$, amiből pedig a triviális fűrészjel:

$$s(n) = 2p(n) - 1 \tag{2.7}$$

A használt polinom ebben az esetben az s^2, amit egyszer deriválunk $s(n)-s(n-1)$ szerint, végül $1/4T_0$ értékkel skálázunk [11, 12].

Az y(n) fűrészjelre $N = 2$ esetben az alábbi általános alakot tudjuk bevezetni:

$$y(n) = \begin{cases} y_A(n) & \text{ha } p(n) < T_0 \\ y_B(n) & \text{ha } p(n) \geq T_0 \end{cases} \tag{2.8}$$

ahol A az átmeneti régió, B a lineáris régió. A fentiek alapján ezek kimeneti jelek kiszámolhatóak.

A lineáris szakaszon $s(n-1) = 2(n-1)T_0 - 1$ és $s(n) = 2nT_0 - 1$, így

$$y_B(n) = \frac{(2nT_0 - 1)^2 - (2(n-1)T_0 - 1)^2}{4T_0} = 2nT_0 - 1 - T_0 \tag{2.9}$$

Az eredményben látható a triviális jel T_0 offsettel. Ez a fél mintavételi idővel való eltolásnak felel meg (hiszen a triviális jel $2T_0$ értékkel nő mintánként), ami a deriválás következménye. Az átmeneti szakaszon az előzőhöz hasonlóan $s(n-1) = 2(n-1)T_0 - 1$, azonban itt 1 fölé érve a modulo művelet miatt $p(n) = nT_0 - 1$, így $s(n) = 2(nT_0 - 1) - 1 = 2nT_0 - 3$.

$$y_A(n) = \frac{(2nT_0 - 3)^2 - (2(n-1)T_0 - 1)^2}{4T_0} = 2nT_0 - 1 - T_0 - 2\left(n - \frac{1}{T_0}\right) \tag{2.10}$$

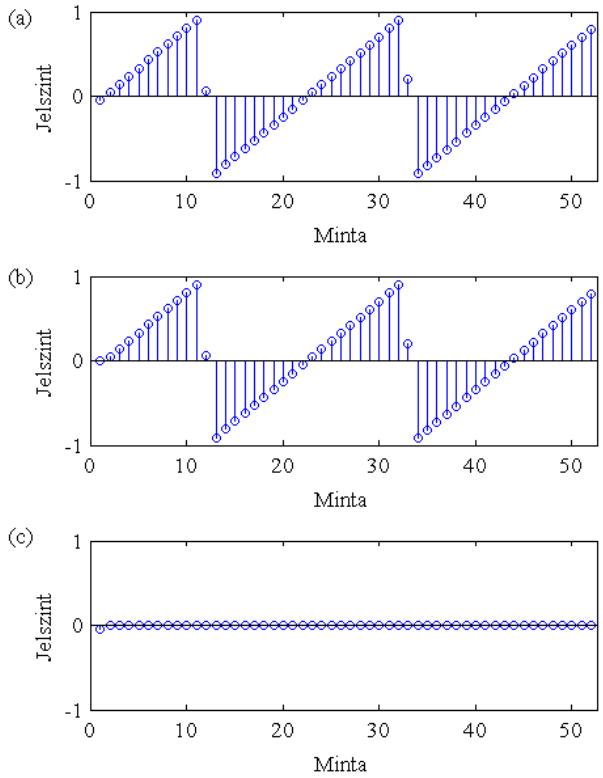
Az $\left(n - \frac{1}{T_0}\right) = D(n)$ kifejezés a szakadás és az azt követő minták távolságát fejezi ki. Látható, hogy $y_A(n) = y_B(n) + c_w(D(n))$, ahol jelen esetben a $c_w(D(n)) = -2D(n)$ polinom felel a fűrészjel csúcsának elsimításáért. Tehát összegzésül a PTR által generált jel a triviális jelnek a fél mintavételi idővel való eltoltja, a törésnél egy polinommal korrigálva.

Az eredményül kapott képletekben furcsának tűnhet az n, elvégre semmi sem korlátozza, még a modulo művelet sem. A gyakorlatban azonban programozási szemszögből az n függ a p-től, hiszen a p alapjelet működtetjük modulo számlálóként, ami már korlátos, és abból következik az n értéke, pl. a fűrészjel esetében $p(n) = nT_0$-ból $n = p/T_0$. Ezt behelyettesítve a fent számolt formulákba és átrendezve kapjuk azt az alakot, melyet az oszcillátor programozása közben felhasználunk. Ez persze az összes PTR módszerrel generált jelre igaz lesz.

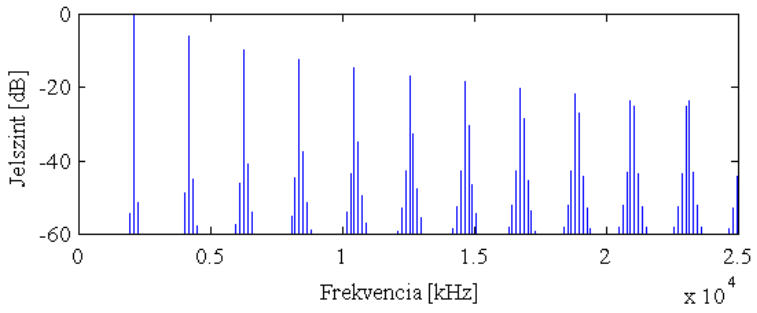
A 2.6 képen látható a PTR és DPW algoritmussal szintetizált jelek. Az első mintában ugyan van egy kis eltérés, mivel a DPW-ben használt deriválásnak szüksége lenne egy előző mintára is ugyanolyan eredményhez, azonban ezen kívül 0 vagy nagyon kicsi értékű különbségfüggvényt tapasztalunk, azaz az algoritmus valóban ugyanaz azt a hullámformát generálja, mint a DPW, csak jelentősen kisebb erőforrás felhasználásával. Így a spektrum is megegyezik, ahogy a 2.7 képen látható.

Négyszögjel

A négyszögjelet a legegyszerűbben a DPW-hez hasonlóan két, egymáshoz képest fél periódussal eltolt fűrészjelből állítható össze. A fűrészjel mintáinak ismeretében könnyedén kiszámolható az egyes szakaszokon a jel. Itt négy régióra oszthatjuk fel a hullámformát:



2.6. ábra. (a) PTR és (b) DPW módszerrel generált jel, illetve (c) különségük



2.7. ábra. PTR módszerrel generált fűrészjel spektruma

$y_A(n)$ Magas értékű lineáris (konstans) szakasz

$y_B(n)$ Lefelé törés környezete

$y_C(n)$ Alacsony értékű lineáris (konstans) szakasz

$y_D(n)$ Felfelé törés környezete

Legyen a két fűrészjel alapjelei p_1, mely 0-ról, és p_2, mely 0,5-ról indulva működik modulo számlálóként. Az ezekből generált hullámformák különbsége lesz a

négyszögjel, azaz $y(n) = y_2(n) - y_1(n)$. Az referenciajelnek a p_1-et tekintjük, p_2-t ehhez fogjuk viszonyítani. A 2.1 táblázatban található a fűrészjelek értéke az egyes régiókban.

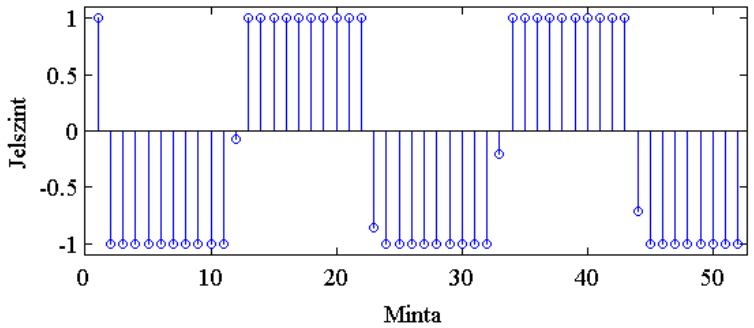
	$y_1(n)$	$y_2(n)$
A	$2nT_0 - 1 - T_0$	$2(nT_0 + 0,5) - 1 - T_0$
B	$2nT_0 - 1 - T_0$	$2(nT_0 - 0,5) - 1 - T_0 - 2(n - 1/T_0)$
C	$2nT_0 - 1 - T_0$	$2(nT_0 - 0,5) - 1 - T_0$
D	$2nT_0 - 1 - T_0 - 2(n - 1/T_0)$	$2(nT_0 + 0,5) - 1 - T_0$

2.1. táblázat. A fűrszjelek az egyes szakaszokon

Innen a $y_2(n) - y_1(n)$ kivnonással könnyen adódik a négyszögjel $y(n)$ függvénye:

$$y(n) = \begin{cases} 1 + 2(n - 1/T_0) & \text{ha } p_1(n) < T_0 \\ 1 & \text{ha } T_0 \leq p_1(n) < 0,5 \\ -1 - 2(n - 1/T_0) & \text{ha } 0,5 \leq p_1(n) < 0,5 + T_0 \\ -1 & \text{ha } 0,5 + T_0 \leq p_1(n) \end{cases} \tag{2.11}$$

A fűrészjelek megfelelő skálázásának köszönhetően a 2.8képen látható jel pontosan -1 és $+1$ között mozog, további skálázásra nincs szükség. Megfigyelhető, hogy ugyanaz a $2(n - 1/T_0)$ polinom korrigálja lefelé és felfelé a jelet, mint a fűrésznél. Természetesen ez a várt viselkedés, elvégre ugyanolyan jellegű, $+1$ és -1 közötti törés simábbá tételére szolgál. Az offset megszűnik a különbségképzésnek hála. Elmondható, hogy az előállítás során jelentősen kisebb az erőforrásigény, elvégre sem két fűrészjel-generátorra, sem további deriválásokra nincs szükség.



2.8. ábra. PTR algoritmussal előállított négyszögjel

2.4. Kiterjesztés PWM alkalmasságára

Több tipikus hullámforma, mint a háromszög és négyszög, nem csak szimmetrikusan, hanem valamilyen $0 < D < 1$ kitöltési tényező szerint is előállítható. Ekkor az

egyik szakasza a jelnek (pl. háromszög növekvő szakasza) D részét teszi ki egy periódusnak, míg a másik értelemszerűen $1 - D$ részét.

Az így előállított hang nem csak, hogy az eddigiektől eltérően szól, szélesítve így a létrehozható hangképek skáláját, de a tényező modulálásával még különlegesebb hangzásokat kapunk eredményül. Ez a moduláció a pulzusszélesség moduláció, azaz pulse width modulation (PWM).

Célunk tehát az eddig implementált algoritmusokat kiterjeszteni ennek kezelésére. A továbbiakban átvesszük, hogyan lehet DPW és PTR estében megvalósítani.

2.4.1. Differentiated Parabolic Waveform

Háromszögjel

Az 50% kitöltési tényezőjű jel a szimmetrikus, a 100%-hoz közeledve pedig a fűrészjelhez hasonlít. Így akár nincs is feltétlenül szükség külön fűrészjel-generátorra, hiszen a háromszögjel pl. 99%-os kitöltési tényező mellett már 440 Hz felett (az A_4 hangtól) megegyezik a fűrészjellel.

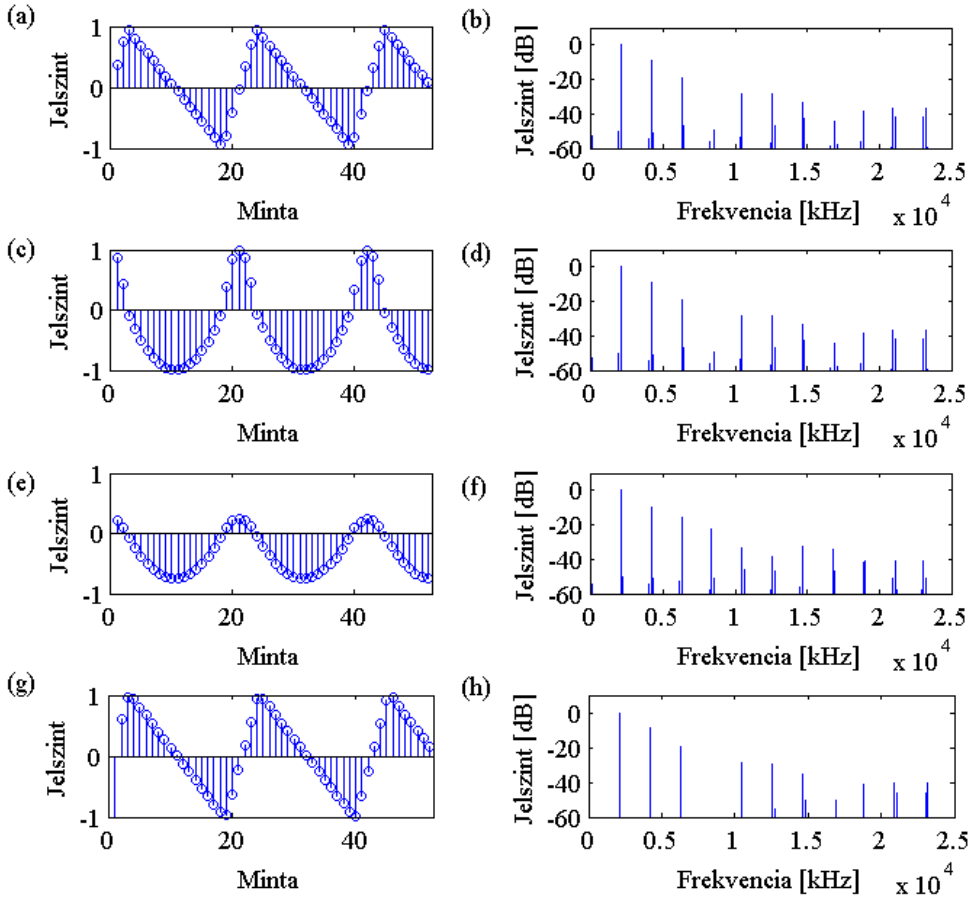
A jel DPW algoritmussal történő előállítása néhány módosítástól eltekintve megegyezik a 2.3.1 fejezetben taglalt szimmetrikus jel generálásával, ennek folyamata a 2.9 ábrán követhető végig. A triviális jelet négyzetre emeljük, majd hasonlóan meg kell szorozni egy négyszögjellel. Azonban így a különböző hosszúságú szakaszokon is 1 a jel maximális abszolútértéke, ahogy a (c) ábrán látható. Így a 0 környékén hirtelen változik a meredekség, ezért a parabolikus jeleket megfelelő aránnyal csökkenteni kell, hogy a jel végig folytonosan deriválható legyen. Tehát a négyszögjel megegyező frekvenciájú és kitöltési tényezőjű, és egyik szakasza D, másik szakasza $1 - D$ értékű, ahol D a kitöltési tényező ($0 < D < 1$). A szorzás eredménye látható a (e) ábrán. Ezután következik a szokásos deriválás.

A triviális jel előállítása során különösen nagy gondot kell fordítani arra, hogy -1 alá vagy $+1$ fölé érve mi legyen a számláló új értéke, ugyanis megváltozik a meredekség. Amennyiben erre nem fordítunk kellő figyelmet, a parabolikus hullámformában a 0 közelében törések lesznek, amik a deriváláskor csúcsokat eredményeznek. A számítások során a következő eredmények adódtak az új értékeknek: (2.12), amikor $+1$ fölé, (2.13), amikor -1 alá ér a számláló.

$$cntr = 1 - (cntr - 1)\frac{D}{1 - D} \tag{2.12}$$

$$cntr = -1 - (cntr + 1)\frac{1 - D}{D} \tag{2.13}$$

Az algoritmusban használt négyszögjel a triviális jellel együtt generálható, az aktuális számlálási irány függvényében kell minden mintának D vagy $1 - D$ értéket



2.9. ábra. Háromszögjel generálása DPW algoritmussal: (a) alapjel, (c) korrigálandó jel, (e) parabolikus jel, (g) végső jel, és ezek spektrumai

adni. A két jel szorzását követően alakul ki a folytonos parabolikus jel, amit deriválva létrejön a kívánt háromszögjel.

Négyszögjel

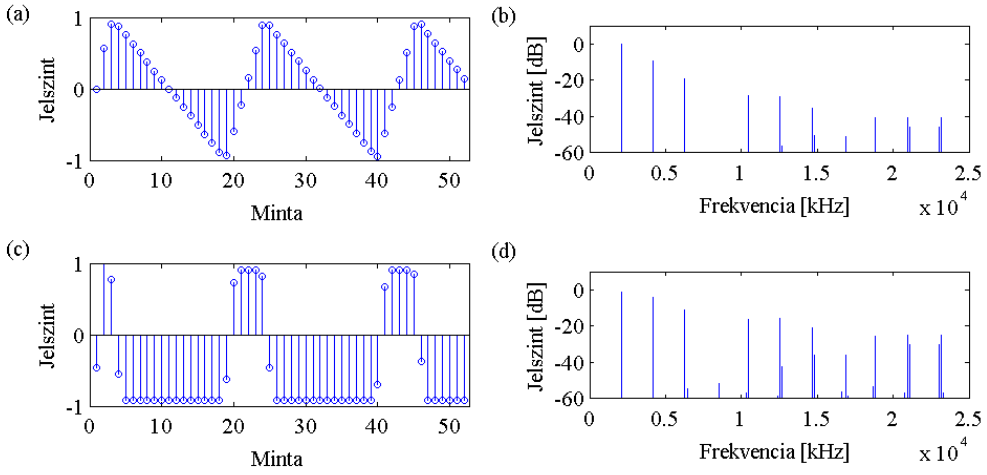
Maga a jelgenerálás a háromszögjel után már viszonylag könnyű: a háromszögjel deriválásával négyszögjelet kapunk, mindössze az amplitúdót kell kompenzálni, hogy a jel -1 és $+1$ között mozogjon.

Ez azonban nem triviális, lévén a háromszögjelben előforduló különböző meredekségek miatt a pozitív és negatív minták abszolút értéke nem fog megegyezni, vagyis a jelnek lesz ofszetje. Ezt az ofszet "hibát" kell kiküszöbölni, majd szorozni egy számmal a frekvencia függvényében. Számítások alapján a jelet $\frac{f}{f_s}\left(\frac{1}{1-D}-\frac{1}{D}\right)$ értékkel kell feljebb tolni.

A szorzáshoz az elméleti számítások eredménye $c = D \cdot (1-D) \cdot f_s/f$, de ez csak közelíti a kívánt eredményt, mivel +1-nél nagyobb értékeket vesz fel a jel. Ki-

csivel ugyan, de 1.0-nál nagyobb amplitúdójú jeleket nem tudunk kezelni, így ezt a hibát kompenzálni kell. A próbálgatások során $c \cdot (1 - f/f_s)^2$ értékkel való szorzás megoldotta a problémát, alacsony frekvenciákon se volt nagyobb a jel abszolút értéke 1-nél. A magasabb frekvenciákon már nagyobb csökkenés észlelhető, de 4 kHz környékén is 1 dB az átlagkülönbség.

Ez a módszer még javítható, ugyanis az ofszet "hiba" kiküszöbölése még nem tökéletes. A pozitív és negatív minták abszolút értéke továbbra sem egyezik meg kellően, és a kompenzáló szorzás is azért lett négyzetes, mivel a $c \cdot (1 - f/f_s)$ szorzás után az egyik szakaszon ugyan az abszolút érték kisebb volt 1-nél, a másikon viszont nem. A hiba korrigálásával talán módosítani lehet a kompenzáló szorzást is, így magasabb frekvenciákon sem fog csökkenni a hangerő.



2.10. ábra. (a) Háromszögjel és (b) spektruma, (c) a deriválással képzett négyszögjel és (d) spektruma

Másik lehetőség, hogy két eltolt fűrészjel különbségét alkalmazzuk. Az eljárás azonos a szimmetrikus négyszögnél látottakkal, a különbség annyi, hogy az ott előírt félperiódus helyett itt tetszőleges az eltolás mértéke [5]. Az egyik jelet a periódus D-ad részével késleltetve a különbség a D hosszú szakaszon $1 - D$, az $1 - D$ hosszú szakaszon pedig $-D$ lesz. Így, hogy a jel -1 és 1 között mozogjon, meg kell szorozni 2-vel, és $-2(1 - D) + 1 = 2D - 1$ offsetet kell hozzáadni.

2.4.2. Polynomial Transition Regions

Négyszögjel

Míg DPW használatával a négyszögjel előállításához szükség volt fűrészre vagy háromszögre, a PTR esetében ez nincs így, sőt, sokkal könnyebb, mint a háromszög, ezért ennek taglalásával kezdjük.

Az aszimmetrikus jelet hasonlóan generáljuk, mint a szimmetrikusat, mindössze a p_2 kezdőállapota változik: 0,5 helyett D. Így D periódusidőnyivel eltolva a két jelet a generált jel szakaszainak aránya $D : 1 - D$ lesz, így a kívánt pulzusszélességeket kapjuk. A megkülönböztethető régiók hasonlóan a magas és alacsony konstans jel, valamint a szakadások környezete, és itt a fűrészjelek értékeit a 2.2 táblázat foglalja össze.

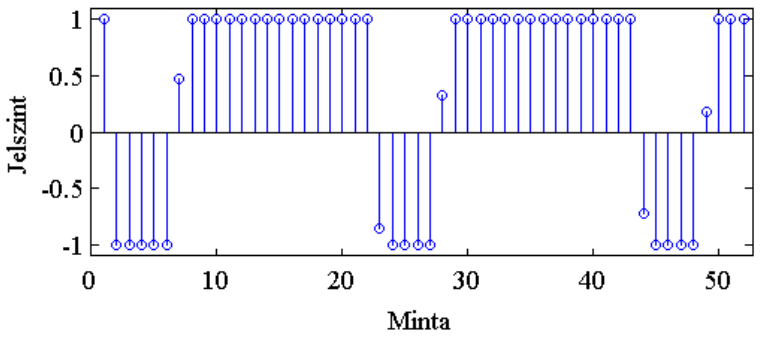
	$y_1(n)$	$y_2(n)$
A	$2nT_0 - 1 - T_0$	$2(nT_0 + D) - 1 - T_0$
B	$2nT_0 - 1 - T_0$	$2(nT_0 - (1 - D)) - 1 - T_0 - 2(n - 1/T_0)$
C	$2nT_0 - 1 - T_0$	$2(nT_0 - (1 - D)) - 1 - T_0$
D	$2nT_0 - 1 - T_0 - 2(n - 1/T_0)$	$2(nT_0 + D) - 1 - T_0$

2.2. táblázat. A fűrészjelek az egyes szakaszokon

Az egyes szakaszokat kiszámolva azt tapasztaljuk, hogy a jel $2D$ és $-2(1 - D)$ között mozog, azaz van offset hibája. Ennek a hibának a kompenzálásához $-2D+1$ offsetet kell a jelre tenni, ezáltal a régiókban az értékek megegyeznek a szimmetrikus jelével, mindössze a határok helyében van különbség. Így a PWM négyszögjel:

$$y(n) = \begin{cases} 1 + 2(n - 1/T_0) & \text{ha } p_1(n) < T_0 \\ 1 & \text{ha } T_0 \leq p(n) < D \\ -1 - 2(n - 1/T_0) & \text{ha } 0,5 \leq p_1(n) < D + T_0 \\ -1 & \text{ha } D + T_0 \leq p(n) \end{cases} \tag{2.14}$$

Az eredmény a 2.11 ábrán látható. A hullámforma tökéletesen megegyezik a DPW algoritmussal előállítottal, így a spektruma is.



2.11. ábra. PTR algoritmussal előállított négyszögjel

Háromszögjel

A legnagyobb kihívást a PTR algoritmus kiterjesztésében a változtatható kitöltési tényezőjű háromszögjel megalkotása jelentette.

A hullámforma most is négy régióra osztható:

$y_A(n)$ Növekvő lineáris szakasz

$y_B(n)$ Maximum csúcs környezete

$y_C(n)$ Csökkenő lineáris szakasz

$y_D(n)$ Minimum csúcs környezete

Legyen a p alapjel a triviális PWM háromszögjel, a 2.4.1 fejezetben taglaltakhoz hasonló módon. Vegyük át a jelgenerálás közben a jel viselkedését az egyes szakaszon.

A növekvő lineáris szakaszon mintánként $2T_0/D$-vel nő a jel. Itt két szomszédos minta $p(n-1) = 2(n-1)T_0/D$ és $p(n) = 2nT_0/D$. Mindkét mintára meghívódik a $1-p^2$ függvény, valamint beszorzódnak $-D$-vel. Így a deriválás $-D(1-p^2(n)) - (-D)(1-p^2(n-1))$ alakban írható fel. Végül a szokásos módon $1/4T_0$ számmal skálázva az eredmény:

$$y_A(n) = \frac{2nT_0}{D} - \frac{T_0}{D} \tag{2.15}$$

A csökkenő lineáris szakaszon mintánként $2T_0/(1-D)$-vel csökken a jel, így két szomszédos minta $p(n-1) = -2(n-1)T_0/(1-D)$ és $p(n) = -2nT_0/(1-D)$. A fentivel megegyező módon számolva a generált hullámforma:

$$y_C(n) = -\frac{2nT_0}{1-D} + \frac{T_0}{1-D} \tag{2.16}$$

A csúcsok esetében bonyolultabb a számolás, elvégre a két minta a háromszögjel különböző régióiban található. Kezdjük a maximum csúccsal. Az első minta itt $p(n-1) = \frac{2(n-1)T_0}{D}$, a második pedig az átmeneten található, így a 2.4.1 fejezetben látható módon $p(n) = 1 - \left(\frac{2nT_0}{D} - 1\right)\frac{D}{1-D}$. Azonban a generálás során, mivel az első minta a növekvő szakaszon van, a második pedig már a csökkenőben, ezért előbbi $-D$, utóbbi pedig $1-D$ értékkel lesz megszorozva. Így a deriválás alakja $(1-D)(1-p^2(n)) - (-D)(1-p^2(n-1))$, amit skálázva:

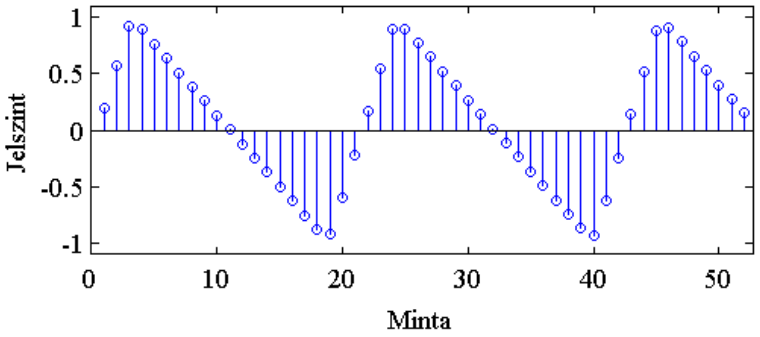
$$y_B(n) = \frac{2nT_0}{D} - \frac{T_0}{D} - \frac{n^2T_0}{D} - \frac{n^2T_0}{1-D} - \frac{D^2}{4(1-D)T_0} - \frac{D}{4T_0} + \frac{n}{1-D} \tag{2.17}$$

Különbség a fűrész- és négyszögjelhez képest, hogy itt az átmeneti régióban található minta értéke n-nek, és így a p alapjelnek másodfokú függvénye. Ez annak

tudható be, hogy a háromszög esetében nincsen szakadása a jelnek, ezért jellegre más, magasabb fokszámú polinom szükséges a minták korrigálásához.

A minimum csúcs környezetében hasonló módon járunk el. Az első $p(n-1) = -\frac{2(n-1)T_0}{1-D}$ minta a csökkenő szakaszon $1-D$ szorzót kap, míg a második $p(n) = -1 - \left(-\frac{2nT_0}{1-D} + 1\right)\frac{1-D}{D}$ minta a növekvő szakaszon $-D$-vel szorzódik. A $-D(1-p^2(n)) - (1-D)(1-p^2(n-1))$ deriválás elvégzésével és a skálázással megkapjuk a generált mintát:

$$y_D(n) = -\frac{2nT_0}{1-D} + \frac{T_0}{1-D} + \frac{n^2T_0}{1-D} + \frac{n^2T_0}{D} + \frac{(1-D)^2}{4DT_0} + \frac{1-D}{4T_0} - \frac{n}{D} \qquad (2.18)$$



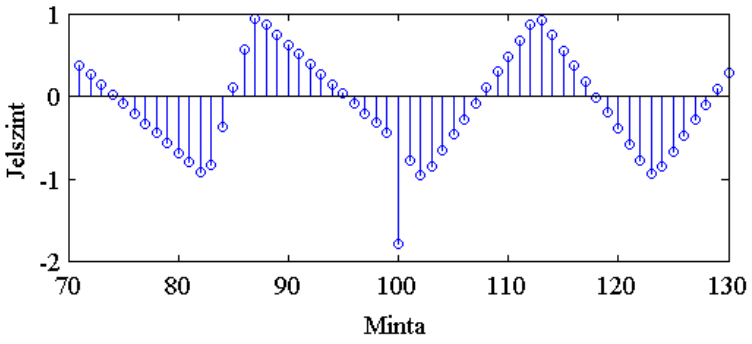
2.12. ábra. PTR algoritmussal generált PWM háromszögjel

Az, hogy az alapjel meredekségében már szerepet játszik a kitöltési tényező, megakadályozza, hogy az eddigi PTR jelekhez hasonló alakban írjuk fel a teljes hullám függvényét. Azt viszont tudjuk, hogy ebben az esetben mindig az az egy minta van az átmeneti régióban, amit akkor generálunk, amikor az alapjel -1 alá vagy $+1$ fölé érne, és az oszcillátor programozásakor ezt használjuk ki az alapjel aktuális értéke helyett.

2.4.3. Kitöltési tényező változásának hatása

A kitöltési tényező modulációját az egyik LFO végzi. Az LFO az oszcillátorokra jellemző hullámformák és szinuszjel generálását használja erre a célra. A VCO oszcillátor paraméterei között található a kitöltési tényező (D) és a modulációs mélység (h). Így az oszcillátor aktuális (adott mintára érvényes) kitöltési tényezőjét a $D \cdot (1 + h \cdot y_{LFO})$ formula adja, ahol y_{LFO} az LFO aktuális kimenete (szintén -1 és $+1$ közötti érték). Értelemszerűen, amikor a modulációs mélység 0 (vagy az LFO ki van kapcsolva), akkor a kitöltési tényező konstans módon a beállított érték.

A DPW esetében ennek a modulációnak meg vannak a korlátai. A jelgeneráló algoritmus érzékeny a pulzusszélesség hirtelen változására, ez a jelben tüskék formájában jelenik meg.



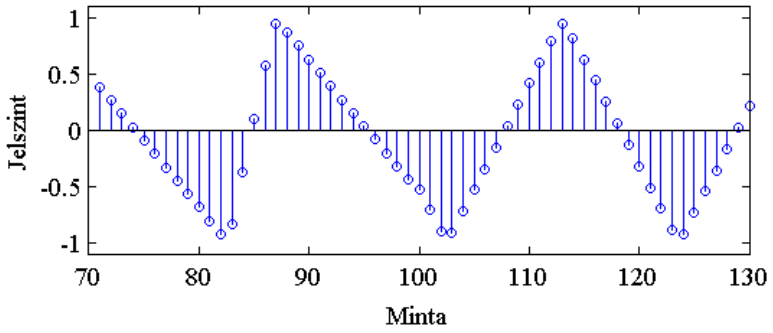
2.13. ábra. A kitöltési tényező hirtelen változásának hibája

A 2.13. ábrán látható példában a 100. mintánál a kitöltési tényező hirtelen 0,2-ről 0,5-re ugrik. Ekkor a parabolikus jelben törés jelenik meg, ami a deriváláskor egy tüskét hoz létre, ami a példában igen nagy, majdnem 2 abszolút értékű. A hangszer játéka közben ez a jelenség kattogásként érzékelhető.

A hiba csökkenthető a kitöltési tényező változásának korlátozásával. A tesztelések során arra lehetett jutni, hogy amikor 44,1 kHz mintavételi frekvencia esetén egy ezreddel (0,001) változhat a modulálójel értéke, a jelgenerálás közben minimális torzulások jelennek csak meg, ezeket pedig hallani már nem lehet. Az LFO maximális frekvenciája is korlátozódik ezáltal, mivel a négyszögjel esetén, ugyan 20 Hz frekvenciánál is van ideje a jelnek -1-ről $+1$-ig eljutni, azonban a jel jellegre jobban hasonlít már a háromszögjelre. Amennyiben a maximális frekvenciát 20 Hz-nél kisebbre választjuk, a jelenség javítható. A hiba a szinuszjellel való moduláláskor nem jelentkezik, a jel jellegének köszönhetően, így azt a szintetizátor jelenlegi formájában is gond nélkül lehet használni.

A hiba okozójaként a deriválást hozhatjuk fel, ezért a PTR alkalmazása közben ügyeltünk ennek elkerülésére. Míg DPW esetben a lineáris szakaszon több műveletet végeztünk, itt mindössze a számlálóhoz adódik egy kis offset, így azt várjuk, hogy a kitöltési tényező változásával csak az alapjel, és így a generált jel meredeksége változik, így tüske nem jelenik meg. Ez így is történik, ahogyan a 2.14 képen látható.

Tehát a PTR módszerrel generált háromszögjel kitöltési tényezője korlátozások nélkül modulálható, így további hangzásokat kelthetünk a szintetizátorunk segítségével.



2.14. ábra. A kitöltési tényező megfelelő változása

2.5. Összehasonlítás

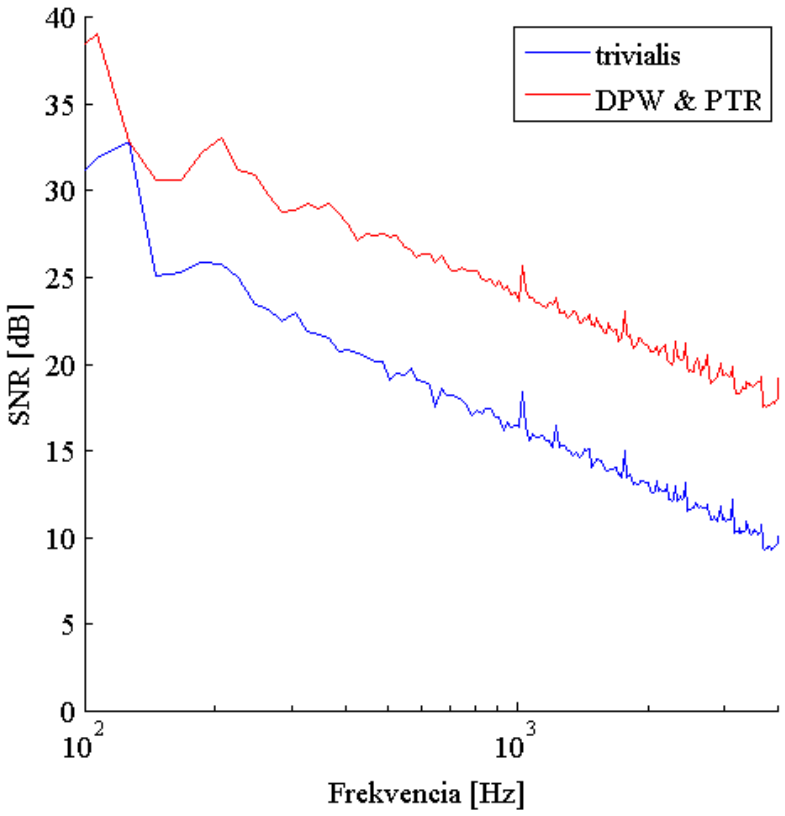
Az eddig tárgyalt algoritmusok közül az ígéretesebbeket két fő szempont, az átlapolódás mértéke és a számításigény alapján összehasonlítjuk, így eldől, melyik a legalkalmasabb a szoftverszintetizátorban való alkalmazásra. A vizsgálandó algoritmusok a két MATLAB-ban implementált módszer, a Differentiated Parabolic Waveforms és a Polynomial Transition Regions. Ehhez a fűrészjel és az aszimmetrikus háromszögjel előállításait vesszük szemügyre.

Az átlapolódást akár hallgatással is lehet tesztelni. Kellően nagy számú tesztalannyal meghallgattatva a különböző algoritmussal generált mintákat felmérhetjük, melyik a legkevésbé zajos, melyik a legkellemesebb a fülnek. Ennél egyszerűbb objektív módszer az aliasing vizsgálatára a jel-zaj viszony (SNR) kiszámítása. Jelnek az additív módon előállított ideális hullámforma spektruma számít, zajnak pedig az átlapolódott komponensek. A számítás módját a 2.19 képlet adja meg.

$$SNR = 10\log_{10}\frac{P_{jel}}{P_{zaj}} = 20\log_{10}\frac{A_{eff,jel}}{A_{eff,zaj}} = 20\log_{10}\frac{\sqrt{\sum\frac{A_{jel}^2}{2}}}{\sqrt{\sum\frac{A_{zaj}^2}{2}}} = 10\log_{10}\frac{\sum A_{jel}^2}{\sum A_{zaj}^2} \tag{2.19}$$

Az additív módszerrel előállított jel szinuszos tagjainak együtthatóit négyzetesen összegezve adódik az ideális jel teljesítménye. A vizsgált algoritmussal előállított jelnek kiszámoljuk a Fourier-transzformáltját, majd az ideális jel spektrumát átskálázzuk úgy, hogy az alapharmonikusa megegyezzen a vizsgált jel alapharmonikusával. A két spektrum különbsége adja az átlapolódott komponensek összességét, és ezek négyzetösszege felel meg a nem kívánt jel teljesítményének [6]. Az 2.19 képlettel az SNR értékeit különböző alapfrekvenciákon is kiszámoljuk, az eredmény a 2.15 ábrán látható.

Természetesen a DPW és PTR algoritmusok átlapolódása megegyezik, mivel



2.15. ábra. A jel-zaj viszony a különböző algoritmusok esetében

ugyanazt a jelet állítják elő. Mindkettő jelentősen felülmúlja a triviális megoldást hangminőség tekintetében.

Így a két megoldás között a számításigény fog dönteni. Azt várjuk, hogy a PTR bizonyul jobbnak, hiszen pont azért fejlesztették ki, hogy a DPW jelét kevesebb számítással is elő tudja állítani. Ennek számszerűsítéséhez azt vizsgáljuk meg, hogy az egyes algoritmusok egy mintát átlagosan mennyi művelet elvégzésével állítanak elő. Ennek összefoglalója a 2.3 táblázatban látható.

A DPW esetében ez konstans, hiszen minden mintán ugyanazokat a műveleteket végezzük el. A fűrészjel alapjelét először megemeljük (összeadás), négyzetre emeljük (szorzás), majd a deriválás során kivonjuk belőle az állapotváltozót (összeadás), végül skálázzuk (szorzás). Így összesen két összeadás és két szorzás szükséges. A háromszögjel ennél csak egy kicsit bonyolultabb: először az alapjelet a növelése után négyzetre emeljük, majd lecsökkentjük (szorzás és két összeadás). Ezután megszorozzuk egy négyszögjellel, deriváljuk (összeadás), végül skálázzuk. Ez összesen három szorzás és két összeadás.

A PTR során a jel különböző szakaszain más műveleteket végzünk. Így egyből meg kell vizsgálni egy elágazást, hogy melyik régiónak megfelelően kell számolni. A lineáris szakaszon $y_B(n) = 2nT_0 - 1 - T_0 = 2p - 1 - T_0$, mivel itt $p(n) = nT_0$, így

az alapjelünkön egy szorzást és egy összeadást végzünk el. Az átmeneti szakaszon $p(n) = nT_0 - 1$, így $y_A(n) = 2p - 1 - T_0 - \frac{2p}{T_0} + 2 = \left(2 - \frac{2}{2T_0}\right) p + 1 - T_0$. Tehát az elvégzendő műveletek száma itt is ugyanannyi, így a teljes szintézis alatt konstans [12].

A PTR algoritmussal generált háromszögjel esetében négy elkülöníthető szakasza van a jelnek, így 3 elágazással lehet eldönteni, hol tart éppen a jel. A növekvő és csökkenő lineáris szakaszokon az alapjelhez csak egy kis offsetet adunk hozzá, azaz egy összadás az összes elvégzendő művelet. Az átmeneti szakaszokon már bonyolultabb a helyzet. A maximumcsúcs esetében a 2.18 képletbe helyettesítve $p(n) = 1 - \left(\frac{2nT_0}{D} - 1\right) \frac{D}{1-D}$ értékét, majd p változójú polinommá alakítva a 2.20 képlet adódik. A négyzetre emelés is szorzás, így az első két taghoz rendre kettő és egy szorzás szükséges, végül ezek összegzéséhez két összeadás. A minimumcsúcs számításához ugyanennyi művelet szükséges szimmetriai okok miatt.

$$y_D(n) = p^2 \frac{D}{4(1-D)T_0} + p \frac{2D(1-2T_0}{4(1-D)T_0} + \frac{D + 4T_0^2 - 4T_0}{4(1-D)T_0} \tag{2.20}$$

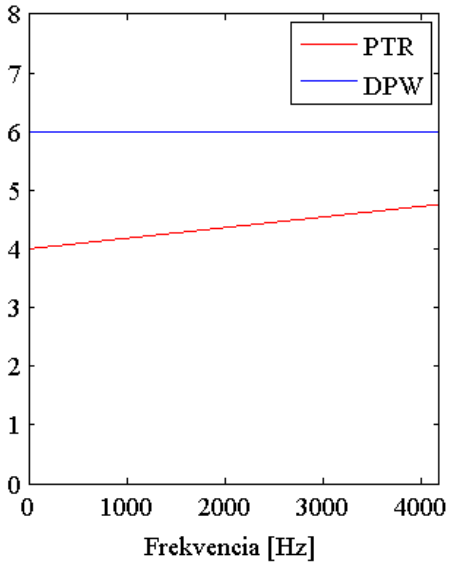
A két csúcsot periódusonként egy-egy mintára kell kiszámolni, a többi mintát mind az egyszerűbb módon. Az, hogy hány ilyen minta van, függ az alapfrekvenciától, ebből következően a számításigény is. Egy periódusban $f_s/f = 1/T_0$ számú minta van. Ebből két minta számolásához szükséges három szorzás és két összeadás, valamint $1/T_0 - 2$ mintához egy összeadás. Így átlagban mintánként $\frac{3 \cdot 2 + 0}{1/T_0} = 6T_0$ szorzásra és $\frac{2 \cdot 2 + 1(1/T_0 - 2)}{1/T_0} = 1 + 2T_0$ összeadásra van szükség. Átlagosan tehát összesen $4 + 8T_0$ műveletet hajt végre a program egy minta kiszámolásához a PTR algoritmussal generált háromszögjel esetében.

Fűrészjel	Összeadás	Szorzás	Elágazás	Összesen
DPW	2	2	0	4
PTR	1	1	1	3
Háromszögjel	Összeadás	Szorzás	Elágazás	Összesen
DPW	3	3	0	6
PTR	$1 + 2T_0$	$6T_0$	3	$4 + 8T_0$

2.3. táblázat. A számításigény fűrészjel és PWM háromszögjel generálása esetén

A fűrészjelnél egyértelműen a PTR igényli a kevesebb erőforrást, mivel az igénye mindkét algoritmussal konstans, azonban a háromszögjelnél a frekvenciafüggés további vizsgálatokat követel. Ábrázoljuk a két algoritmus elvégzendő műveleteinek számát a frekvencia függvényében a 88 billentyű legmagasabb hangjáig, azaz a 4186 Hz-es C_8-ig.

A 2.16 ábrán látható, hogy még a C_8 hang frekvenciáján is kb. 20%-kal kisebb a számításigény, mint a DPW esetében. További számításokkal beláthatjuk, hogy az erőforrásigény 11000 Hz körül lesz azonos, amit ugyan a MIDI még éppen tud



2.16. ábra. A DPW és PTR algoritmusok számításigénye a frekvencia függvényében

kezelni, de gyakorlatban már nem jellemző. (A 11175.3 Hz-es hangnak a 125-ös MIDI-kód felel meg. A maximális kód a 127, hozzá a 12543.8 Hz-es hangmagasság tartozik. A MIDI-vel bővebben a 5.3 fejezetben fogunk foglalkozni.)

Ne feledkezzünk meg arról a tényről sem, hogy a PWM csak a PTR algoritmussal előállított jelek esetében működik korlátozások nélkül. Mivel az így előállított hangzások egészen különlegesek, ez már önmagában jelentős érv a PTR algoritmus mellett.

Összegzésül tehát az átlapolódás mértéke nem döntő szempont, a számításigény és a PWM hatásának vizsgálata pedig azt az eredményt hozta, hogy a Polynomial Transition Regions alkalmasabb a gyakorlati használatra, így a VST szintetizátor programozásakor ezt az algoritmust fogjuk felhasználni.

3. fejezet

VCF

3.1. Moog szűrő modellje

A szintetizátorban a Minimoog szűrőjének digitális modelljét implementáljuk. Az analóg Moog szűrő 4 sorba kapcsolt elsőfokú aluláteresztő szűrőt tartalmazott, valamint egy globális negatív visszacsatolást annak érdekében, hogy a törési frekvencia közelében rezonancia jelenjen meg. A rezonancia mértéke szabályozható volt, akárcsak az elsőfokú szűrők, és így a teljes rendszer töréspontí frekvenciája. Ez a két paraméter egymástól független volt. A szűrő továbbá képes a sajátrezgésre, azaz tulajdonképpen oszcillátorként is használható [5].

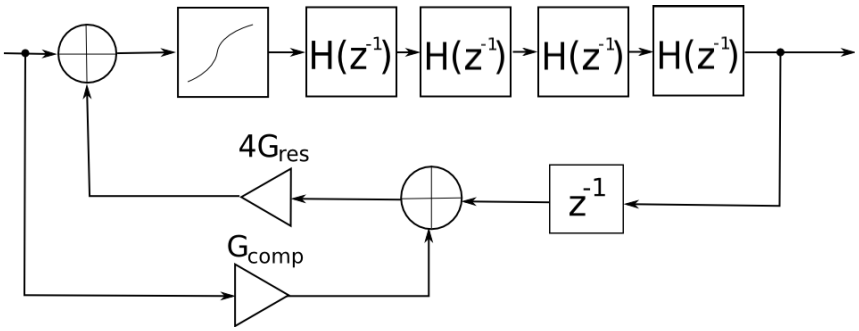
A digitális modell is hasonlóan 4 sorba kapcsolt elsőfokú szűrőből áll visszacsatolással, a visszacsatoló ágban egy késleltetéssel. Ez a késleltetés azért szükséges, mert nélküle az aktuális kimeneti jel kiszámításához szükség lenne az éppen számítandó értékre, ez pedig nem megvalósítható. Ezzel azonban megszűnik a töréspontí frekvencia és rezonancia paraméterek függetlensége. Ez nem szerencsés, hiszen az egyik módosításával a másik nem kívánt értékűre áll át. Ugyan létezik a teljes szűrőnek olyan implementációja is, mely megszünteti a visszacsatolóágban a késleltetést a szűrő átvitelének explicit kiszámításával [13], azonban viszonylagos bonyolultsága miatt nem ezt alkalmaztam.

Több módszert vizsgáltak az elsőfokú szűrő digitális modellezésére, amivel a két paraméter függetleníthető. A bilineáris és hátratartó differencia transzformációkkal kapott közelítések még nem adtak megfelelő eredményt. Végül a 3.1 átviteli függvénnyel rendelkező, bizonyult a legjobbnak. A $z = -0,3$ értékű zérus közel teljesen függetleníti egymástól a két paramétert [14].

$$H(z^{-1}) = \frac{g + g \cdot 0,3z^{-1}}{1,3 + 1,3(g-1)z^{-1}} \tag{3.1}$$

A g paraméter határozza meg az elsőfokú alulátersztő töréspontí frekvenciáját. Alacsony frekvenciákon ez meghatározható a $g \approx 2\pi f_c/f_s$ módon, azonban

magasabb frekvenciákon a törésponti frekvencia így egyre inkább eltér az f_c kívánt értéktől. A $z = -0,3$ zérus és a visszacsatolóágban lévő késleltetés miatt a kívánt törésponti frekvenciához a g-t bonyolultabb számításokkal lehet csak kiszámolni. Egy negyedfokú kompenzáló polinomot használva adódik a következő kifejezés: $g = 0.9892\omega_c - 0.4342\omega_c^2 + 0.1382\omega_c^3 - 0.0202\omega_c^4$, ahol $\omega_c = 2\pi f_c/f_s$ [5]. Az így megkapható törésponti frekvencia megfelelő közelítést ad a kívánt frekvenciára.



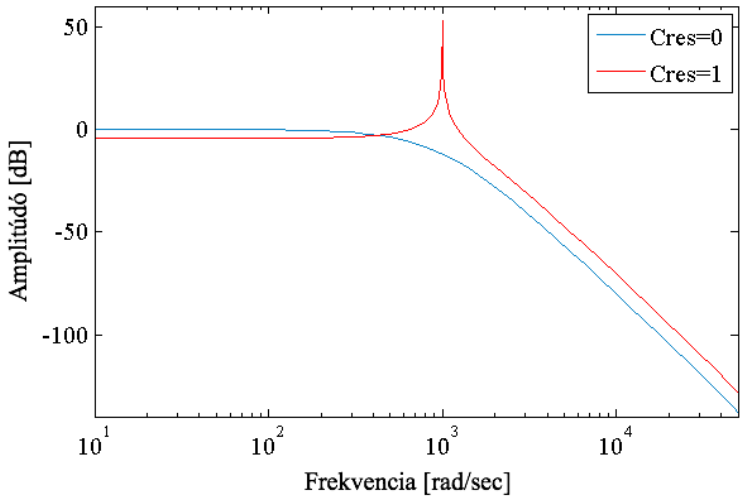
3.1. ábra. A teljes szűrő

A teljes szűrő modellje a 3.1. ábrán látható [5]. A 4 elsőfokú szűrőn és a visszacsatolásban lévő késleltetésen kívül egyéb elemeket is észrevehetünk.

A visszacsatoló ágban a C_{res} paraméterrel lehetne a rezonancia mértékét beállítani, azonban így a visszacsatolás is csak alacsony frekvenciákon képes közelíteni a kívánt értéket. Ehelyett egy G_{res} paramétert csatolunk a rendszerhez, ami szintén egy kompenzáló polinomként írható fel: $G_{res} = C_{res} \cdot (1.0029 + 0.0526\omega_c - 0.0926\omega_c^2 + 0.0218\omega_c^3)$ [5].

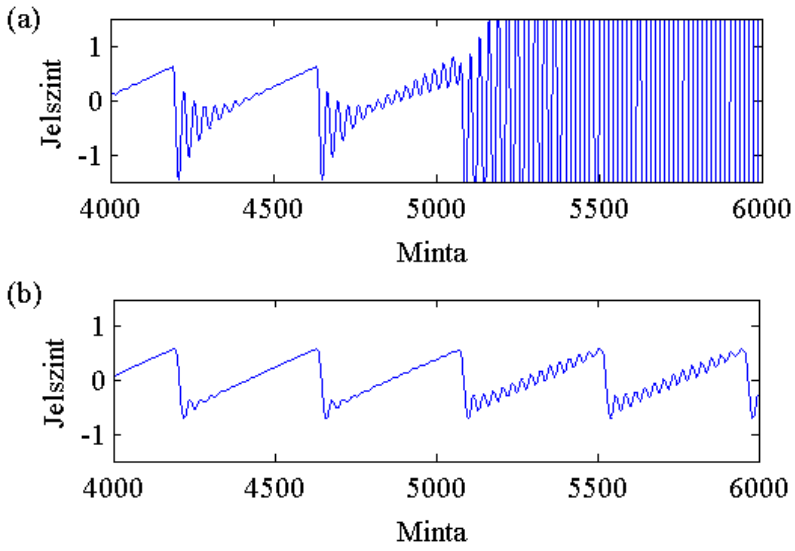
A rezonancia paramétert 0-ról 1-re növelve az áteresztő tartomány erősítése 12 dB-lel csökkenne. Ennek kompenzálására szolgál a G_{comp}, aminek 1,0 értéke azonban a rezonanciafrekvencia környezetében túl nagy erősítést eredményezne. Kompromisszumos megoldásként legyen $G_{comp} = 0,5$, így $C_{res} = 1,0$ esetén is az áteresztő tartományban a csökkenés mindössze 6 dB, és a rezonanciafrekvencia körül is elfogadható a karakterisztika [5]. A 3.2. ábrán a teljes szűrő átvitele látható a $C_{res} = 0$ és 1 értékekkel.

Az eredeti Moog szűrő nemlineáris volt, azonban pont ez adta meg a sajátos hangzását, így ezt a hatást érdemes szimulálni. Az eredeti analóg szűrő legjobb közelítésében minden elsőfokú szűrőbe egy nemlineáris tag kapcsolódik, amely a bemeneti jelének hiperbolikus tangensét adja ki. Ennek a megoldásnak előnye volt még a közeli hangzás mellett, hogy képes volt a sajátrezgésre [15]. A hátránya ennek a nagy számításigény, hiszen mintánként négyszer kell a tanh függvényt meghívni. Ennek redukálása érdekében csak egy nemlineáris tag található a rendszerben, sorba kapcsolva a kaszkád elsőfokú szűrőkkel. A hangzás ugyan jobban eltér így az eredeti szűrőétől, azonban csökkent az erőforrásigény, továbbá a sajátrezgés képessége és a korlátos kimenet megmaradt [5].



3.2. ábra. A szűrő átvitele a rezonancia függvényében

A 3.3 ábra demonstrálja a nemlinearitás szükségességét. Nélküle a rezonancia paraméter 1-nél nagyobb értékei esetén a rezgő jel "elszáll" a végtelenbe. A korlátos tanh tagot beillesztve a rendszerbe a sajátrezgés során is korlátos marad a hullámforma.

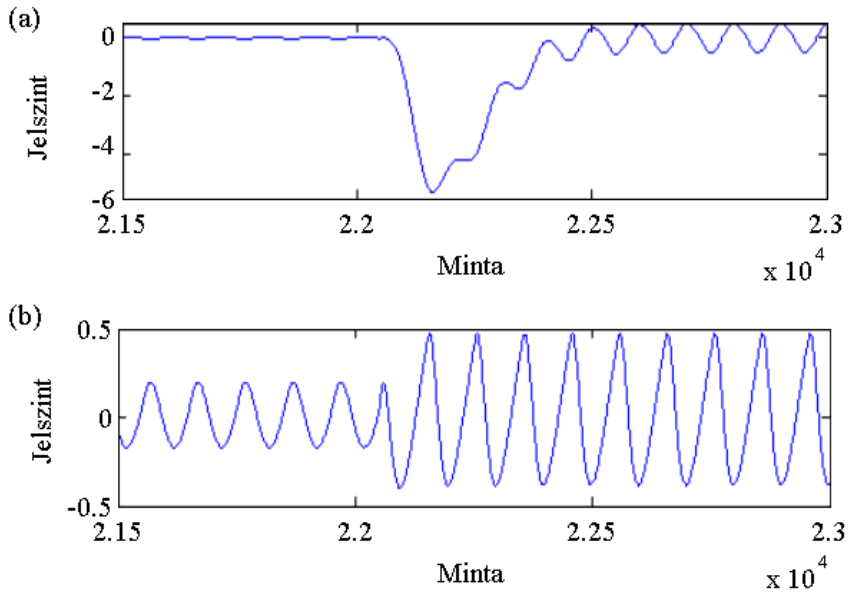


3.3. ábra. $C_{res} = 1.1$ (a) nemlinearitás nélkül és (b) a tanh függvénnyel

3.2. Töréspontí frekvencia modulálása

Magasabb fokú szűrőket szokás első- vagy másodfokú szűrők sorba kapcsolásával realizálni. Ennek előnye, hogy valamelyik paraméter változásakor a tranziensek sokkal

kevésbé jelentősek. A Moog szűrő esetében is a sorba kapcsolt változat jobban kezeli a törésponti frekvencia paraméter hirtelen változását, mint egyetlen negyedfokú szűrő [13]. A 3.4. (a) ábra mutatja a negyedfokú szűrős rendszer válaszát, amikor a törésponti frekvencia 500 Hz-ről 1000 Hz-re ugrik. Hatalmas és hosszú idejű (kb. 500 mintáig tartó) tranzienssel kell számolni, míg a sorba kapcsolt elsőfokú szűrők esetében ez a tranziens nagyon rövid idejű, nem jelentős. (A 3.4. ábrán a skálázás ugyan eltérő, de a két jel nagysága a tranziensen kívül megegyezik.)



3.4. ábra. (a) Negyedfokú szűrő és (b) az implementált kaszkád szűrő reakciója a törésponti frekvencia hirtelen változására

Tehát ezzel a megoldással a törésponti frekvencia az LFO-val gond nélkül modulálható, még hirtelen váltásokkal is. Arra viszont ügyelni kell, hogy még a modulált törésponti frekvencia sem lépheti át a mintavételi frekvencia felét, azaz 22050 Hz-et. A számítás módja $f_c = f_{c,VCF}(1+y_{LFO})$, ahol $f_{c,VCF}$ a szűrőn beállított, f_c a valódi törésponti frekvencia, y_{LFO} pedig az LFO kimenete ($-1 \leq y_{LFO} \leq 1$). $1 + y_{LFO}$ maximálsi értéke 2 lehet, ezért ahhoz, hogy f_c legnagyobb lehetséges értéke 22050 Hz legyen, $f_{c,VCF}$ maximumát $f_c/2 = 11025$ Hz-ben kell meghatározni.

4. fejezet

Kiegészítő egységek

4.1. ADSR

A jel szűrése után már csak a jel erősítése van hátra. Mivel a digitális modellünkben a jel értéke a -1...+1 tartományban mozog, és a szintetizátor kimenetére is ugyanekkora jel kerül, az analóg szintetizátorokban lévő előerősítőhöz hasonló eszközre nincs szükség. Így a virtuális hangszerben a VCA maga a burkológörbe-generátor lesz.

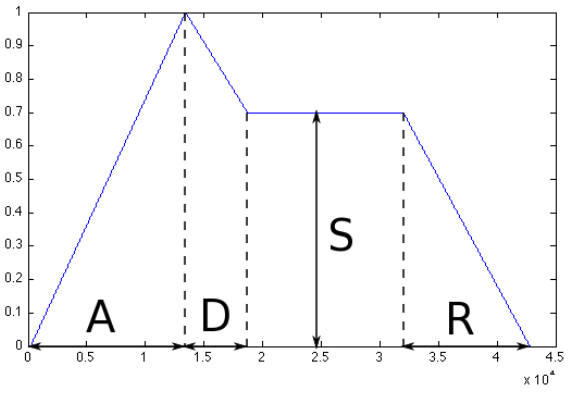
Burkológörbe-generátoroknak is több változata létezik, a fejlesztés során ezek közül az ADSR típus került implementálásra. Az egység paraméteri:

Attack idő, amely a billentyű lenyomásától a maximális hangerő eléréséig telik el

Decay idő, amely a hangerő meghatározott értékére való csökkenése alatt telik el

Sustain jelszint, melyet a billentyű nyomva tartása alatt ad ki a hangszer

Release idő, amely a hang teljes elcsendesedéséig telik el a billentyű felengedése után



4.1. ábra. Az ADSR burkológörbe-generátor

A burkológörbe-generátor egy állapotgéppel valósítható meg. Egy billentyű lenyomásakor az ADSR *attack* állapotba vált, majd a paraméternek megfelelő idő alatt a kimenet 0-ról 1-re növekszik. A növekedés lineáris, a számítás pedig lineáris interpolációval történik. Két minta értéke közötti különbség $(f_s \cdot A_{max} \cdot A)^{-1}$, ahol A_{max} a beállítható maximális időtartam másodpercben mérve (a mi szintetizátorunkban ez 5 másodperc), A pedig a 0 és 1 közötti vezérelhető paraméter. Ez abból adódik, hogy $A_{max} \cdot A$ másodpercben $f_s \cdot A_{max} \cdot A$ minta alatt kell 0-ról 1-re növekednie. Amikor eléri az 1 értéket, *decay* állapotba vált, ami alatt szintén interpolációval működik, csak a jel 1 és a *sunstain* érték között csökken. *Sustain* állapotban konstans érték van a kimeneten egészen addig, amíg a billentyű elengedése *release* állapotba nem teszi az ADSR-t, amiben a jel a megadott időtartam alatt lecsökken 0-ra.

Szokás még ADSR egységekkel a szűrőt is vezérelni, ebben a modellben azonban csak a hangerőt szabályozza.

4.2. LFO

Az LFO szinte megegyezik a VCO-val. Olyan alacsony frekvenciájú (általában maximum 20 Hz-es) hullámformákat generál a kimenetére, mint pl. fűrészjel, háromszögjel, négyszögjel, valamint a szinuszjel, ami a VCO-kra a felharmonikusok hiánya miatt nem jellemző. A jelek generálása megoldható triviális módon is, nincs szükség a szakadásokhoz köthető nagyfrekvenciás komponenseket kiszűrni, mivel a 2.4.3. és 3.2. fejezetekben tapasztaltak szerint a PTR algoritmussal generált hullámforma kitöltési tényezőjének és a töréspontí frekvenciának hirtelen változása nem okoz problémát.

Amire a VCO és LFO programozása során ügyelni kell, hogy mivel a VCO-ban külön beállítható a kitöltési tényező, amit az LFO még modulál is, előfordulhat, hogy a kitöltési tényező eléri a 0 vagy 1 értéket, ami 0-val való osztást eredményez, valamint 1-nél magasabb értéket is felvehet. Ezért korlátozni kell a moduláció mélységét egy $b = c \cdot (0,5 - |0,5 - D_{VCO}|)$ értékkel, ahol $c < 1$, de 1-hez közeli (mi esetünkben $c = 0,99$), D_{VCO} a VCO-n beállított kitöltési tényező. Így a valódi $D = D_{VCO} + b \cdot y_{LFO}$ sosem éri el (és lépi át) a 0 vagy 1 értékeket.

A szinuszjelnél arról érdemes szót ejteni, hogy a sin() függvény meghívása minden egyes mintához jelentősen növeli a számításigényt. Ez különféle módszerekkel csökkenthető, pl. lineáris interpoláció alkalmazásával, azaz adott számú mintánként számítjuk ki a pontos értéket, és minden tovább minta a két ilyen mintát összekötő egyenesen található. Az implementáció során minden 100. minta kerül kiszámításra, így a kapott jel a szinuszjelet jól közelíti, és a számításigény jelentősen csökken.

A virtuális analóg szintetizátor implementációja MATLAB környezetben tehát teljes:

- Az oszcillátor egy nyers alapjelet generál, melynek pulzusszélessége modulálható
- Az alapjelet a szűrő megszűri, kialakítva a hangzást, mely a törésponti frekvencia modulálásával tovább színesíthető
- Végül a burkológörbe-generátor meghatározza, hogy a hangerő a billentyűk kezelésekor hogyan viselkedjen

A továbbiakban az így megszerzett ismereteket átültetjük VST környezetbe.

5. fejezet

Megvalósítás VST környezetben

5.1. VST

A Virtual Studio Technology (VST) a német Steinberg zenei technológiával foglalkozó cég által fejlesztett interfész, mellyel szoftveres hangszerek és effektek könnyedén integrálhatóak zenei szerkesztő szoftverekbe [16]. Léteznek direkt a VST-k kezelésére készített szoftverek, valamint ma már szinte mindegyik zenekészítő program (Digital Audio Workstation – DAW) támogatja a használatukat. Ennek is köszönhetően széles körben elterjedt, rengeteg VST hangszer és effekt érhető el fizetős és ingyenes verzióban egyaránt, de főként csak Windows platformra, az egyéb platformokra való nehézkés fejlesztés miatt, valamint OS X rendszereken az Apple saját hasonló programja, az Audio Unit az elterjedtebb.

A VST-ket megkülönböztetjük feladatuk szerint, két fő kategória a VST hangszerek (VSTi) és a VST effektek. A VST hangszerek hangot generálnak, és MIDI üzeneteket dolgoznak fel a működésük során. A VST effektek audiojeleket dolgoznak fel, MIDI üzeneteket szintén tudnak fogadni.

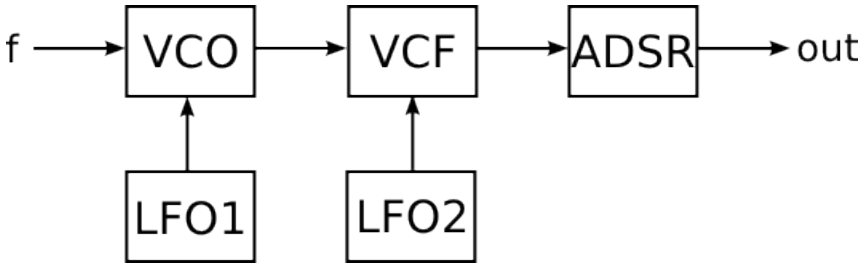
A Steinberg szabadon elérhetővé tette a VST SDK csomagját, mely bárki számára lehetővé teszi VST-k fejlesztését. Az SDK azokat a könyvtári függvényeket tartalmazza, amelyekkel valamely fejlesztőkörnyezet segítségével egy VTS C++ nyelven megírható. A 2.0 verzió dokumentációja részletes és egyszerű példákon keresztül rengeteg segítséget nyújt a fejlesztés során.

A virtuális analóg szintetizátor VST írását Code::Blocks fejlesztőkörnyezetben végeztem C++ nyelven a VST SDK 2.4-es verziójának segítségével.

5.2. A szintetizátor leprogramozása

A C++ objektumorientált programozás. A szintetizátor különböző egységeit érdemes ezért osztályokként megvalósítani, így az egységek egy-egy példányának össze-

kötése fogja adni a működőképes szoftverszintetizátort.



5.1. ábra. A program blokkdiagramja

A 5.1 képen látható a megvalósított szintetizátor blokkdiagramja. Mindegyik blokk a különböző osztályok egy példánya. LFO-ból van kettő is, mivel az oszcillátor és a szűrű paramétereit külön-külön moduláljuk. Ezeken felül létezik egy szintetizátor osztály, mely elvégzi a VST hosttal való kommunikációt, és vezérli a belső egységeket. Minden osztálynak friend osztálya lesz a szintetizátor osztály, így az el tudja érni bármelyiknek a változóit. Most tekintsük át röviden az egyes osztályokat.

A VCO osztály kimenetén mintánként adja ki a PTR algoritmussal generált alapjelet (lsd. 2.3.2. fejezet) a jel típusa (fűrész, négyszög vagy háromszög), frekvenciája és kitöltési tényezője függvényében. A kitöltési tényező modulálható az első LFO-val. Az LFO osztály is oszcillátorként működik, a különbség mindössze annyi a VCO-hoz képest, hogy csak szimmetrikus háromszög és négyszögjeleket tud előállítani, helyette szinuszgörbére képes.

A VCF osztálynak két bemenete van, a VCO-ból és a második LFO-ból kijövő minták, a kimenetén pedig kiadja a szűrt jelet. Vezérelhető paraméterei az LFO-val modulálható töréspontí frekvencia, valamint a rezonancia mértéke.

Az ADSR osztály a burkológörbe-generátort valósítja meg. Amennyire egyszerű az elmélete, annyira fontos szerepe van a programozott szintetizátorban, mivel a billentyűk lenyomásával és elengedésével ezt az egységet vezéreljük. Lenyomáskor belép az *attack*, elengedéskor pedig a *release* állapotba. Ezáltal meghatározza, hogy a bemenetére érkező szűrt jel meg is szólaljon-e, illetve ha igen, milyen hangerővel.

A kisebb egységeket áttekintettük, már csak az egész szintetizátor működése igényli, hogy néhány szót szóljunk róla. A szintetizátor osztályának mint a hangszer vázának meg kell felelnie pár speciális követelmények, hogy VST-ként működhessen. Az osztály létrehozását az AudioEffectX leszármazottjaként és konstruktorban szükséges függvényhívásokról az SDK dokumentációja részletesen beszél [17]. A konstruktoron belül jönnek létre az egységek példányai. Utána megkezdődik a szintetizátor fő feladata, a hang generálása.

A *processReplacing(float** inputs, float** outputs, VstInt32 sampleFrames);* függvény ciklikusan hívódik meg a működés során, minden egyes hívás után több mintát

generál [17]. Az *inputs* paraméterre értelemszerűen nincs szükség, hiszen a hangszernek nincs bemenete. A függvény megkapja, milyen hosszú mintákból álló tömböt kell létrehoznia, és ezt milyen címre tegye. A megadott tömböt mintáról mintára feltölti az egységek függvényeinek meghívásával. Egy példa a függvény használatára:

```
void MinimoogishSynth::processReplacing (float** inputs,
float** outputs, VstInt32 sampleFrames)
{
    float* out1 = outputs[0];
    float* out2 = outputs[1];

    float oscOut, fltrOut, adsrOut;

    while (--sampleFrames >= 0)
    {
        oscOut = VCO1.process(LFO1.proccess());
        fltrOut = VCF1.filter4pole(oscOut, LFO2.process());
        adsrOut = ADSR1.process() * fltrOut;
        (*out1++) = (*out2++) = adsrOut;
    }
}
```

A programban azonban ebben a formában nem lehet megadni, milyen magasságú hang szóljon, ráadásul folyamatosan szól a bekapcsolást követően. Ezeket a paramétereket MIDI üzenetekkel tudjuk vezérelni.

5.3. MIDI kezelése

A szintetizátornak MIDI üzenetekkel tudunk utasításokat adni. Az elkészült program jelenleg csak azokat az üzeneteket tudja feldolgozni, amelyek egy billentyű lenyomásáról és elengedéséről ad információt. Egy MIDI üzenet felépítése [17, 18]:

státuszbájt Egy szám a 0x80 – 0xFF tartományban. 0xEF-ig a második négy bájt a csatornát jelöli – nekünk csak egy csatornánk van, így azt figyelmen kívül tudjuk hagyni. A szám az esemény jellegét jelöli. Az elengedés (note OFF) a 0x80, a lenyomás (note ON) a 0x90 kódnak felel meg.

1. **adatbájt** Note ON és OFF esetében ebből az adatból olvasható ki, melyik billentyű lett lenyomva vagy elengedve. 0-127 intervallumban lévő szám, pl. a 69 a 440 Hz-es A_4 hangnak felel meg [4].

2. **adatbájt** Note ON és OFF esetében a billentyű lenyomásának sebességét jelenti. 0-127 tartományban lévő szám. Jelenleg a program nem dolgozza fel.

A MIDI üzeneteket a host küldi a VST-nek. Az üzenetek VstMidiEvent nevű struktúrák, melyek többek között tartalmazzák a 3 előzőleg tárgyalt bájtot (midiData[4] – a 4. bájt nem használatos), és a jelenlegi VST által feldolgozandó blokk elejéhez képest az esemény bekövetkeztének idejét (deltaFrames).

Az üzenetek feldolgozását a processEvents(VstEvents* events) függvény végzi. Minden ciklusban meghívódik a processReplacing() függvény előtt, így a jelgenerálás a már feldolgozott MIDI üzenetek alapján tud történni [17].

```
VstInt32 MinimoogishSynth::processEvents (VstEvents* events)
{
    VCO1.currentDeltaOn = VCO1.currentDeltaOff = -1;

    for (long i = 0; i < events->numEvents; i++)
    {
        if ((events->events[i])->type == kVstMidiType)
        {
            VstMidiEvent* event = (VstMidiEvent*)events->events[i];
            char* midiData = event->midiData;
            VstInt32 status = midiData[0] & 0xf0;
            if (status == 0x90 || status == 0x80)
            {
                VstInt32 note = midiData[1] & 0x7f;
                if (status == 0x80)
                    noteOff(note, event->deltaFrames);
                else
                    noteOn(note, event->deltaFrames);
            }

        }
    }
    return 1;
}
```

A feldolgozás során az events vektorba összegyűjtött eseményeket sorban megvizsgáljuk. A midiData első bájtjából kiszűri a csatornát, majd összehasonlítja a note ON és note OFF kódjával. Amennyiben valamelyikkel megegyezik, betölti az első adatbájtot is, amely a billentyűt jelöli. Ezután annak függvényében, hogy note ON vagy OFF üzenet érkezett, meghív egy megfelelő függvényt, amelynek átadja a leütött/felengedett billentyűt, és a deltaFrames értéket.

```
void MinimoogishSynth::noteOn(VstInt32 note, VstInt32 delta)
{
    float noteFreq = freqtab[note] / (float)freqScaler;
    VCO1.freq = noteFreq;
    VCO1.currentDeltaOn = delta;
```

```
}
void MinimoogishSynth::noteOff(VstInt32 note, VstInt32 delta)
{
    float noteFreq = freqtab[note] / (float)freqScaler;
    if (VCO1.freq == noteFreq)
    {
        VCO1.currentDeltaOff = delta;
    }
}
```

Note ON esetében nincs szükség külön vizsgálatra, beállítja a VCO frekvenciáját a billentyűnek megfelelőre, valamint a deltaFrames számot beírja a VCO currentDeltaOn változójába, és újraindítja az ADSR-t.

Note OFF esetén azonban szeretnénk, hogy csak a hang csendesedjen el, amelyiket elengedtük. Amikor az egyik billentyűt még nyomjuk, és lenyomunk egy másikat is, az új hang kezd el szólni, a régebben lenyomott billentyű elengedésekor azonban nem szabad a jelgenerálásnak leállnia. Ezért, mielőtt a függvény átállítani az ADSR-t release állapotba, meg kell vizsgálni, hogy az oszcillátor aktuális hangja megegyezik-e a felengedett billentyűvel. Ha nem, a program nem csinál semmit, ha igen, az esemény idejét beírja az oszcillátor currentDeltaOff változójába.

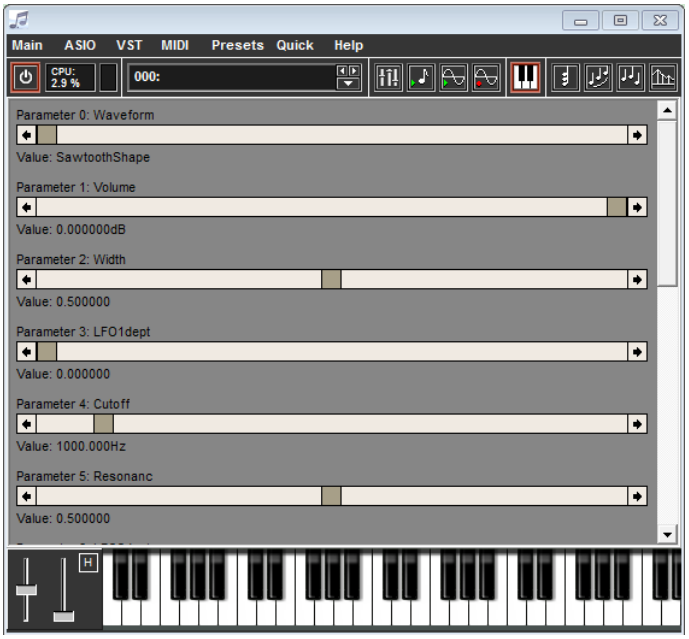
```
void MinimoogishSynth::processReplacing (float** inputs,
float** outputs, VstInt32 sampleFrames)
{
    float* out1 = outputs[0];
    float* out2 = outputs[1];

    while (--sampleFrames >= 0)
    {
        if (sampleFrames == VCO1.currentDeltaOn)
        {
            VCO1.start();
        }
        if (sampleFrames == VCO1.currentDeltaOff)
        {
            VCO1.stop();
        }

        (*out1++) = (*out2++) = VCO1.process();
    }
}
```

A VCO currentDeltaOn és currentDeltaOff változóit ezután a processReplacing() dolgozza fel. Ezek az előállítandó blokkban lévő helyzetet, azaz az esemény idejét jelenti. A függvény ezeket a sampleFrames, azaz a blokkban hátralévő minták számával összehasonlítva tudja a megfelelő időben a megfelelő eseményt előidézni.

A program lényeges blokkjait áttekintettük, a teljes forráskód a mellékletben található.

5.4. Értékelés

Az elkészített program jelenlegi verziója a 5.2. ábrán látható. A VST a Minihost nevű, ingyenesen elérhető host programba van integrálva.



5.2. ábra. A hangszer

A tesztelések során várt viselkedést tapasztaltunk. A hangok megfelelő magasságon szólaltak meg. A Polynomial Transition Regions jelgeneráló algoritmus implementálása sikeres volt, az alaphangok zajmentesen szólnak. A paraméterek állítgatása is működött, az oszcillátor jelalakjának és a szűrő töresponti frekvenciájának, rezonancia-paraméterének módosítása megfelelően változtatott a hangszínen. Az LFO-k bekapcsolása esetén tisztán hallható volt a frekvenciának és a hullámformának megfelelő moduláció. A burkológörbe-generátor a beállított időtartamok alatt hangosította fel a hangot a billentyű lenyomása után, és halkította le az elengedést követően. A VST tökéletesen működik külső MIDI-billentyűzettel is.

6. fejezet

Összefoglalás és fejlesztési lehetőségek

A szakdolgozatban egy virtuális analóg szintetizátort valósítottam meg. Az ilyen szoftverekre és hardverekre manapság nagy az igény, mivel megfelelő minőséggel tudnak a drágán beszerezhető, klasszikus analóg szintetizátorokhoz hasonló hangzást előállítani, ráadásul olcsóak, így bárki számára elérhetőek. A szintetizátort az egyik legnépszerűbb szoftveres formában, VST pluginként implementáltam.

Az első fejezetben rövid történeti áttekintés után átvettük a szubtraktív szintézis elvén működő hangszerek belső egységeit, a jelgeneráló oszcillátort (VCO) és a szűrőt (VCF), valamint a kiegészítő egységeket, az alacsony frekvenciás moduláló oszcillátort (LFO), és az ADSR típusú burkológörbe-generátort. A szintetizátor szerkezetének ismeretében specifikáltuk a megvalósítandó VST plugin képességeit.

A második fejezetben a szakdolgozat fő témáját, a VCO jelgenerálási módszereit tárgyaltuk. Bemutattuk, hogy egy hullámforma triviális előállításakor, azaz amikor a generált jel a mintavételezett analóg hullámformának felel meg, nagymértékű az átlapolódás, és ez gyenge hangminőséghez vezet. Röviden összefoglaltuk az additív szintézis, valamint a kvázi-sávkorlátozott algoritmusok (BLIT, BLIT-FDF BLEP, MinBLEP, PolyBLEP) lényegét. Bővebben azokat az algoritmusokat tárgyaltuk, melyek a mintavételezni kívánt jel spektrumának meredekségét módosították. A Differentiated Parabolic Waveform módszer elve az, hogy az analóg jel integrálásával a spektruma meredekebben csökken, így ekkor mintavételezve az átlapolódás kisebb mértékű, ezután diszkrét időben deriválva pedig visszakapjuk a generálandó hullámformát. Átvettük a fűrészjel és a szimmetrikus háromszögjel generálásának módját, valamint a négyszögjel előállítását két eltolt fűrészjel különbsége és a háromszögjel deriválása segítségével. Előbbi módszerrel a négyszögjel kitöltési tényezője változtatható az eltolás mértékének állításával. Végül a Polynomial Transition Regions algoritmust tárgyaltuk, mely ugyanazt a jelet állítja elő, mint a DPW, kisebb számításigénnyel. Az elv az, hogy mivel a DPW periódusonként egy mintát változtat

meg, ezért elég ezt az egy mintát korrigálni, a jel többi szakaszán pedig számlálóval lehet meghatározni az értékeket. Ez jelentősen kevesebb számolással jár, hiszen a DPW módszer minden egyes mintán elvégezte az integrálást és a deriválást. Mindkét algoritmust kiterjesztettem PWM háromszögjel előállítására, a kitöltési tényezőt az első LFO-val lehet modulálni. Végül a két algoritmust összehasonlítottuk átlapolódás és számításigény szempontjából. Mivel a generált hullámformák megegyeznek, a hang minősége is azonos, erőforrásigény szerint pedig a PTR bizonyult jobbnak. A VST plugin programozásakor így ezt az algoritmust használtam fel.

A harmadik fejezetben a szintetizátor szűrőjét vázoltuk fel, amely a klasszikus Moog szűrőt modellezi. Négy sorba kapcsolt elsőfokú szűrőből áll, késleltető visszacsatolással és egy nemlinearitással kiegészítve. A visszacsatolás paraméterének állításával befolyásolható a rezonanica mértéke, 1 feletti értéknél pedig a nemlinearitás korlátozza (az egyébként végtelenbe elszálló) jelet, különleges hangzást eredményezve. A törésponti frekvencia a második LFO-val modulálható, és láthattuk, hogy ebben a sorba kapcsolt rendszerben a törésponti frekvencia hirtelen változtatása nem okoz problémát, csak kis tranzienseket eredményez, míg egy negyedfokú szűrővel a hosszú és nagy tranziensek gondot jelentenének.

A negyedik fejezetben bemutattuk a kiegészítő egységeket, a burkológörbe-generátort és az LFO-t. A burkológörbe-generátor a kimeneti jel hangerejét szabályozza, így a billentyűk lenyomásával ezt az egység vezérelhető: lenyomáskor felhangosítja a hangot, elengedéskor pedig elnémítja. A megvalósításhoz a lineáris ADSR típust választottam. Az egység az egyes állapotokban lineáris interpolációval számolja ki a burkológörbe értékeit. Az LFO alacsony frekvenciájú hullámformákat generál, amiket a többi egység egyes paramétereinek modulálására használunk. Az előállított jelek a mintavételezett analóg jelnek felelnek meg, valamint a szinuszjelet lineáris interpolációval állítja elő.

Az elmélet után az implementáció következett. Összefoglaltuk a VST pluginek, valamint a fejlesztéséhez nyújtott ingyenesen elérhető csomag, a VST SDK főbb tulajdonságait. A szoftvert C++ nyelven írtam, a szintetizátor egyes egységeit osztályokként valósítottam meg, ezeket pedig megfelelő függvények meghívásával kötöttem össze. Bemutattuk, hogy a VST a generált mintákat egy speciális függvénnyel kezeli, valamint hogyan lehet egy billentyű lenyomását és elengedését jelző MIDI üzeneteket feldolgozni. A kész program megfelelt az előírt specifikációnak. A VST plugin és a forráskódok megtalálhatóak a CD-mellékleten.

A szakdolgozatban új eredmény a Differentiated Parabolic Waveform és Polynomial Transition Regions algoritmusok alkalmassá tétele változtatható kitöltési tényezőjű háromszögjel előállítására. Az átlapolódás a vártnak megfelelően kisebb, mint a triviálisan előállított háromszögjel esetében. A két módszer ugyanazt a jelet állítja elő, számításigény szempontjából a PTR alkalmazása kedvezőbb. Összeha-

sonlítottuk a két jelet továbbá a kitöltési tényező hirtelen változásának hatása szempontjából. A tapasztalatok szerint a DPW a deriválás művelet miatt erre érzékeny módszer, mivel a jelben tüskék jelentek meg, míg a PTR esetében nem jelentkezik ez a probléma.

A hangszer további funkciókkal bővíthető az egyszerűbb kezelhetőség és a hangszínek szélesebb palettája érdekében:

- MIDI-támogatás bővítése:
  - Billentyű leütésének sebessége, azaz a velocity kezelése.
  - További MIDI üzenetek kezelése, így a paraméterek egy MIDI kontroller potmétereivel is vezérelhetőek lennének.
- A hangszer polifonikussá tétele, azaz egyszerre több hang meg tud szólalni. Mivel a processzor jelenlegi terheltsége 3-4% körül mozog egy hang generálásakor, a polifónia megvalósítása nem okozhat nagy gondot.
- Szűrő vezérlése ADSR-rel.
- Effektek: delay, reverb, chorus, stb.
- További új funkciók, pl. arpeggiator.

Irodalomjegyzék

[1] Minimoog: Listening challenge: Software vs hardware synths. http://www.musicradar.com/news/tech/listening-challenge-software-vs-hardware-synths-140104.

[2] Arturia minimoog v vst download. http://www.muno.pl/news/arturia-minimoog-v-for-free-vst-download/.

[3] Synthesizer. http://en.wikipedia.org/wiki/Synthesizer.

[4] Note names, midi numbers and frequencies. http://www.phys.unsw.edu.au/jw/notes.html.

[5] Vesa Välimäki and Antti Huovilainen. Oscillator and filter algorithms for virtual analog synthesis. *Computer Music Journal*, 30(2):19–31, 2006.

[6] Vesa Välimäki. Discrete-time synthesis of the sawtooth waveform with reduced aliasing. *Signal Processing Letters, IEEE*, 12(3):214–217, March 2005.

[7] Vesa Välimäki and Antti Huovilainen. Antialiasing oscillators in subtractive synthesis. *Signal Processing Magazine, IEEE*, 24(2):116–125, March 2007.

[8] J. Nam, V. Valimaki, J.S. Abel, and J.O. Smith. Efficient antialiasing oscillator algorithms using low-order fractional delay filters. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(4):773–785, May 2010.

[9] Eli Brandt. Hard sync without aliasing. In *Proceedings of the International Computer Music Conference*, pages 365–368, 2001.

[10] J. Pekonen, J. Nam, J.O. Smith, and V. Välimäki. Optimized polynomial spline basis function design for quasi-bandlimited classical waveform synthesis. *Signal Processing Letters, IEEE*, 19(3):159–162, March 2012.

[11] Jari Kleimola and Vesa Välimäki. Alias-suppressed oscillators based on differentiated polynomial waveforms. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(4):786–798, May 2010.

[12] Jari Kleimola and Vesa Välimäki. Reducing aliasing from synthetic audio signals using polynomial transition regions. *Signal Processing Letters, IEEE*, 19(2):67–70, February 2012.

[13] Federico Fontana. Preserving the structure of the Moog VCF in the digital domain. In *Proceedings of the International Computer Music Conference*, pages 291–294, 2007.

[14] Tim Stilson and Julius Smith. Analyzing the Moog VCF with considerations for digiral implementation. In *Proceedings of the 1996 International Computer Music Conference*, pages 332–335, 1996.

[15] Antti Huovilainen. Nonlinear digital implementation of the Moog ladder filter. In *Proceedings of the International Conference on Digital Audio Effects*, pages 61–64, 2004.

[16] Virtual studio technology. `http://en.wikipedia.org/wiki/Virtual_Studio_Technology`.

[17] Steinberg Media Technologies GmbH. *Steinberg Virtual Studio Technology Plug-In Specification 2.0 Software Development Kit Documentation*, 1999.

[18] Midi messages. `http://www.cs.cf.ac.uk/Dave/Multimedia/node158.html`.