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

Hatékony oszcillátor algoritmusok analóg szintetizátor modellezésére

TDK dolgozat

2013/14 I. félév

Készítette
Ambrits Dániel
M.Sc. I. évf.

Konzulens
Dr. Bank Balázs

2013. október 25.

Tartalomjegyzék

Kivonat

A 60-as, 70-es években elterjedt analóg szintetizátorok meghatározó szerepet töltöttek be az elektronikus zenében, és egyedi hangzásvilágaiknak köszönhetően ma is töretlen népszerűségnek örvendenek. Azonban ezek az analóg áramkörök segítségével hangot generáló eszközök már csak nehezen vagy drágán érhetőek el, így felmerül az igény a klasszikus hangzások digitális emulálására, akár hardver szintetizátor, akár számítógépen futtatható szoftver formájában. A legelterjedtebb hangszerek a szubtraktív szintézis elvén, azaz egy széles spektrumú alapjel szűrésén alapultak. Az alapjel általában valamilyen egyszerű periodikus jel volt, mint pl. fűrészjel, háromszögjel vagy négyszögjel, amelyet az oszcillátor állított elő. Hogy a kívánt hangzás kialakuljon, a kapott jelet egy analóg szűrő dolgozta fel, végül egy erősítőn keresztül jutott ki a kimenetre. Az egységek paramétereit a zenész közvetlenül, vagy kisfrekvenciás oszcillátorok segítségével vezérelhette.

A dolgozat az első komponens, az oszcillátor modellezésére koncentrál. Az alapjel triviális diszkrét előállítása, azaz az analóg jel mintavételezése a széles spektrum miatt jelentős átlapolódáshoz, így kellemetlen hangzáshoz vezet. A szakirodalom több jelgeneráló algoritmust is bemutat, melyek célja, hogy az eredeti hangzást közelítő sávkorlátozott jelet állítsanak elő. Az egyik legegyszerűbben alkalmazható ilyen módszer a Differentiated Polynomial Waveform algoritmus, amely az eredeti analóg jel integráljának mintavételezésén alapul. A módszer alapelvét felhasználva született meg a Polynomial Transition Regions (PTR) algoritmus, mely a számításigényt jelentősen csökkentette.

A dolgozatban a PTR módszer továbbfejlesztését mutatom be, ami ugyanazt a jelet állítja elő, a számításigényt azonban az összehasonlítások alapján akár további 30%-kal csökkenti. Így a javasolt módszer a jelenleg leghatékonyabb sávkorlátozott oszcillátor algoritmusnak tekinthető. Emellett a módszert változtatható szimmetriájú háromszögjel előállítására is kiterjesztettem, így a szimmetria modulálásával újabb hangzások érhetőek el.

Abstract

Analog synthesizers which have spread in the 60s and the 70s played an important role in electronic music, and they are still popular because of their unique sound. However, these instruments using analog circuits are hard or expensive to purchase, thus there is a need to digitally emulate classic sound, either as a hardware synthesizer or as a software instrument. The most popular instruments are based on subtractive synthesis, which consists in the filtering of a signal having a wide spectrum. This signal is usually a simple periodic waveform like sawtooth, triangle or square, and it is generated by an oscillator. In order to obtain the desired sound, the oscillator signal is processed by an analog filter, then finally it goes through an amplifier. The parameters of each block are controlled by low-frequency oscillators or by the musician directly.

This work focuses on the modeling of the first block, that is, the oscillator. The trivial generation of the discrete-time signal corresponds to the sampling of the analog signal and leads to aliasing and bad sound quality. There are several algorithms presented in the literature generating band-limited signals with reduced aliasing. One of the simplest is the Differentiated Polynomial Waveform (DPW) algorithm which is based on sampling the integral of the original analog signal. The Polynomial Transition Region (PTR) algorithm is the improved version of the DPW method with reduced computational complexity.

This work presents a more efficient version of the PTR algorithm which produces the same signal while requires 30% less operations, making the proposed method the most efficient band-limited oscillator algorithm to date. Moreover, the method is extended to produce asymmetric triangle signals, allowing the creation of new sounds by modulating the symmetry of the waveform.

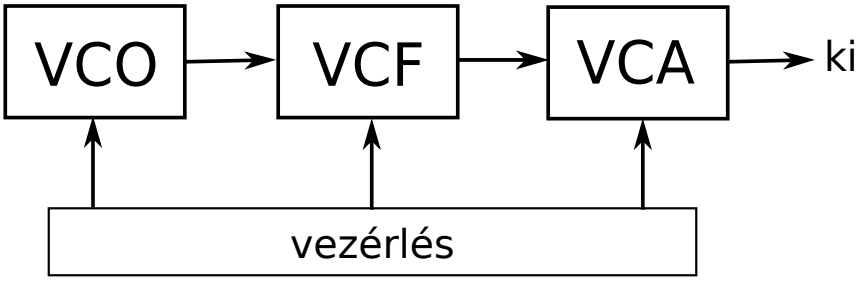
1. fejezet

Bevezető

A 20. század zenei történetében nagy szerepet játszik a szintetizátorok, azaz mesterséges hangok előállítására alkalmas elektronikus eszközök fejlődése [1]. 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 vele 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, előbb hardver, 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 az analóg szintetizátorok a reneszánszukat élik. Azonban ma már ezeket a klasszikus modelleket nem gyártják, egy-egy használt darab pedig elég drágán szerezhető be. Újabban újra jelennek meg analóg szintetizátorok, azonban ezek sem az alsó árkategóriákba tartoznak. Igény van tehát olcsóbb megoldásokra, melyekkel hasonló hangzások állíthatóak elő. Kaphatóak ún. virtuális analóg szintetizátorok is, melyek megjelenése 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 klasszikus modelljeik szoftveres verzióját, természetesen az eredeti hardvernél jelentősen alacsonyabb áron. A szoftverszintetizátorok közül számos ingyenesen elérhető.

A különböző szintetizátorok különféle módszerekkel állították elő a hangot. A főbb kategóriák között található a szubtraktív szintézis, az additív szintézis, FM alapú szintézis vagy a hangminta alapú. Az analóg szintetizátorok túlnyomóan a viszonylag egyszerű szubtraktív szintézis elveit használták. A hangkeltés elve egy harmonikusokban gazdag hullámforma szűrése, ennek megfelelően a hangszer három fő építőeleme egy oszcillátor, egy szűrő és egy erősítő, ez látható a 1.1. ábrán.



1.1. ábra. A szubtraktív szintetizátor váza.

Voltage-controlled oscillator (VCO): A feszültségvezérelt oszcillátor állítja elő a széles spektrumú alapjelet. Az erre a célra leggyakrabban használt hullámformák a fűrészjel, a szimmetrikus háromszögjel és a szimmetrikus négyszögjel. A vezérlést a leütött billentyű alapján kapja, ami a jel (és így a hang) frekvenciáját határozza meg. További paraméter lehet az aszimmetrikus jelek kitöltési tényezője.

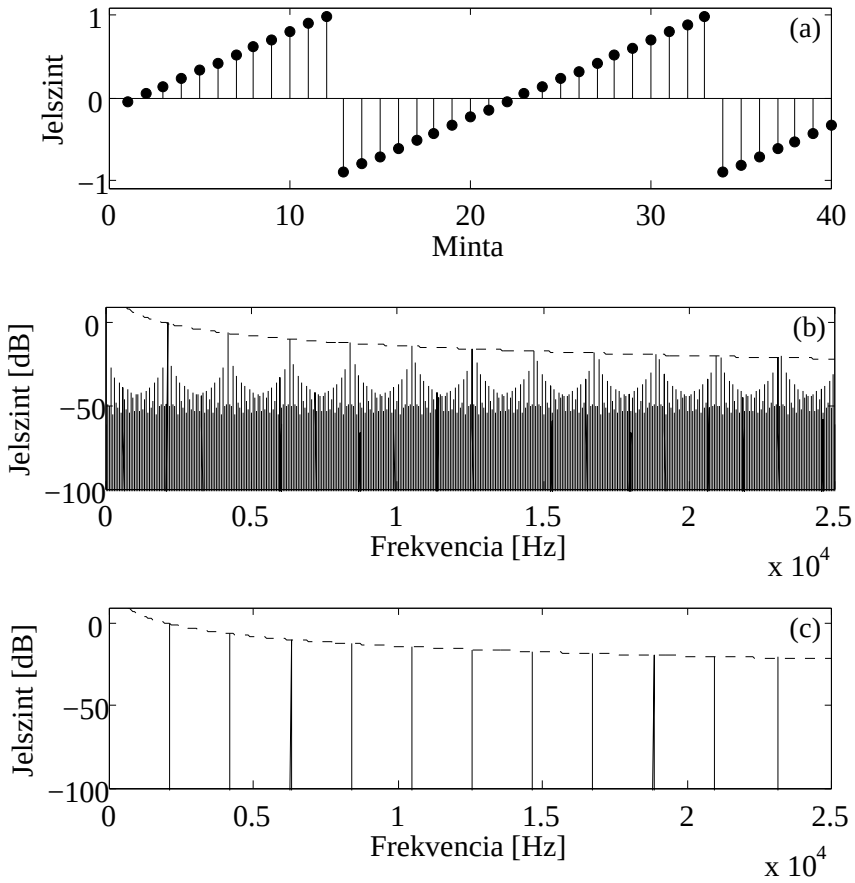
Voltage-controlled filter (VCF): A harmonikusokban gazdag jelet a feszültségvezérelt szűrő dolgozza fel. Leggyakoribb típusa az aluláteresztő szűrő, de léteznek felül-, és sáváteresztő változatok is. Általánosan vezérelhető paraméterei a törésponti frekvencia és a rezonancia mértéke.

Voltage-controlled amplifier (VCA): A feszültségvezérelt erősítő a hangerőt szabályozza.

Ez a váz kiegészülhet további elemekkel. Gyakran az oszcillátor és a szűrő paraméterei kisfrekvenciás oszcillátorok (low-frequency oscillator - LFO) segítségével vezérelhetőek. A 20 Hz-nél kisebb frekvenciájú jelgenerátor gyakori hullámformái között szerepel a fűrész-, háromszög-, négyszögjel, szűrt zaj és a szinuszjel. További kiegészítők lehetnek a különböző burkológörbe generátorok (köztük a leggyakoribb, az Attack Decay Sustain Release, röviden ADSR), melyek a szűrő és az erősítő paramétereit befolyásolhatják. Az ADSR pl. az erősítőn keresztül a hangerőt szabályozza, amikor a billentyűt lenyomjuk vagy elengedjük.

A dolgozat az első egység, azaz az oszcillátor modellezésére koncentrál. A cél, hogy a digitális oszcillátor elő tudja állítani a szokásos periodikus jeleket úgy, hogy a spektrális tulajdonságok minél jobban közelítsék az analóg jelét.

A digitális jel előállításának triviális módja az analóg jel mintavételezése. Ennek gyakorlati megvalósítása egy számláló, amely pl. fűrészjel esetében periodikusan -1-től $+1$-ig felfelé számol [2]. Amikor a jel értéke $+1$ fölé ér, 2-vel lejjebb ugrik, hogy továbbra is analóg jelnek megfelelő hullámformán maradjon.



1.2. ábra. (a) Az analóg jel mintavételezésével generált jel és (b) spektruma. Az analóg jel (c) spektrumával összehasonlítva jelentős a nem kívánt komponensek száma.

A mintavételezés hatására a jel spektruma a mintavételezési frekvenciának megfelelően periodikus lesz. A fűrészjel és a többi használt periodikus jel spektrumai végtelen szélesek, így nem felelnek meg a Nyquist-kritériumnak. Az így átlapolódott komponensek összeadódnak, így a spektrumban jelentős a nem kívánt harmonikusok mennyisége, amely kellemetlen hangzást okoz. A jelenség a 1.2. ábrán látható.

Ezért különböző jelgeneráló algoritmusokat fejlesztettek ki, hogy az analóg jel spektrumát kis számításigénnyel megfelelő mértékben közelítse, minimalizálva az átlapolódás hatását. A következő fejezetben ezen módszerek áttekintése található.

2. fejezet

Korábbi oszcillátor algoritmusok

A szakirodalomban megjelent oszcillátor algoritmusokat három fő csoportba lehet osztani [3]:

1. adott számú harmonikust előállító, sávkorlátozott módszerek
2. a kvázi-sávkorlátozott algoritmusok, melyek az aluláteresztővel szűrt analóg jelet mintavételezik
3. a spektrum meredekségét módosító algoritmusok.

A dolgozat a harmadik kategóriát fogja részletesen tárgyalni, mivel az új módszer is ebbe tartozik.

2.1. Sávkorlátozott módszerek

Az első kategóriába tartozik az additív szintézis. Optimális megoldásnak tűnik, hogy a jelet a megfelelő szinuszos komponensek összegzésével állítjuk elő. A Fourier-sor alapján $f_s/2$ frekvenciáig adjuk össze a komponenseket, így a jel teljesen sávkorlátozott lesz, nincs átlapolódás. Azonban a módszer sok számítást igényel, főként kis frekvenciákon, hiszen $\frac{f_s}{2f}$ darab szinusz értéket kell kiszámolni és összeadni [4]. Mint később látni fogjuk, sokkal kevesebb számítással is elérhető elfogadható eredmény.

Az additív szintézis megvalósítható úgy is, hogy előre kiszámoljuk a jel mintáit különböző számú harmonikusok esetén, és egy táblázatban (wavetable) eltároljuk őket [5]. Ez a wavetable szintézis. Ez a módszer azonban nagy memóriát igényel és a frekvencia változtatásával interpolációra van szükség. A memóriával spórolni lehet, ha pl. oktávonként egy táblázatot mentünk el, amiket különböző sebességgel játszunk le.

2.2. Kvázi-sávkorlátozott algoritmusok

A kvázi-sávkorlátozott algoritmusok a képzeletbeli analóg jelben a Nyquist-frekvencia feletti komponenseket szűrik, így ezt a jelet mintavételezve közel átlapolódásmentes spektrumú jel hozható létre. Az oszcillátorokban használt egyszerű hullámformák esetén analitikusan kiszámolható a szűrt jel.

Ezek között az első a Bandlimited Impulse Trains (BLIT, sávkorlátozott impulzussorozatok) [6]. A módszer egy Dirac-impulzusokból álló folytonos jelből álltja elő a kívánt hullámformákat, majd aluláteresztővel szűri. Impulzussorozat esetén ez azt jelenti, hogy az egyes impulzusok és az átlapolódásgátló szűrő impulzusválaszának konvolúcióját vesszük, azaz az egyes impulzusokat kicseréljük az impulzusválaszra és összegezzük őket. Speciális esete a BLIT-FDF (fractional delay filter), amely a sávkorlátozott impulzusokat törtrészkésleltetővel valósítja meg [7].

A Bandlimited Step Sequences (BLEP) algoritmus a jelben található ugrások sávkorlátozásával állítja elő a kívánt jelet [8]. Az ugrások végtelen meredekségűek, így a spektrumuk is végtelen. A kívánt jel előállításakor a triviális jel ugrásait "kicseréljük" a sávkorlátozott ugrásra. Ennek érdekében az alapjelhez hozzáadódik a BLEP maradék függvény, ami a korlátozott és nem korlátozott ugrás különbsége. Nehézség, hogy az ugráshoz tartozó korrekció a táblázat közepén található, így már előre keresni kell a helyét, hogy az ugrás előtti mintákat is korrigálni lehessen a táblázatban lévő értékekkel. Ezt a problémát küszöbölte ki a MinBLEP [8], ahol minimálfázisú szűrő használatával az impulzus aszimmetrikus lesz, és a táblázat első értéke tartozik az ugráshoz.

A Polynomial Bandlimited Step Function (PolyBLEP) a BLEP algoritmus egyszerűsítése [3]. Az ugrás környezetében található minták módosítására nem egy táblázatban foglalt értékeket használ, hanem zárt alakban megadott egyszerű polinomfüggvényeket. A függvény a szakadás előtti és utáni egy-egy mintát korrigálja. Nagyobb fokszámú polinomokkal természetesen további minták is korrigálhatóak, így pontosabb eredményt kaphatunk, de ennek ára a megnövekedett számításigény. Az eredmény tovább javítható egy spline bázisfüggvényeket alkalmazó optimalizációs eljárással is [9].

2.3. Spektrum meredekségét módosító algoritmusok

2.3.1. A Differentiated Polynomial Waveform algoritmus

Elsőfokú eset

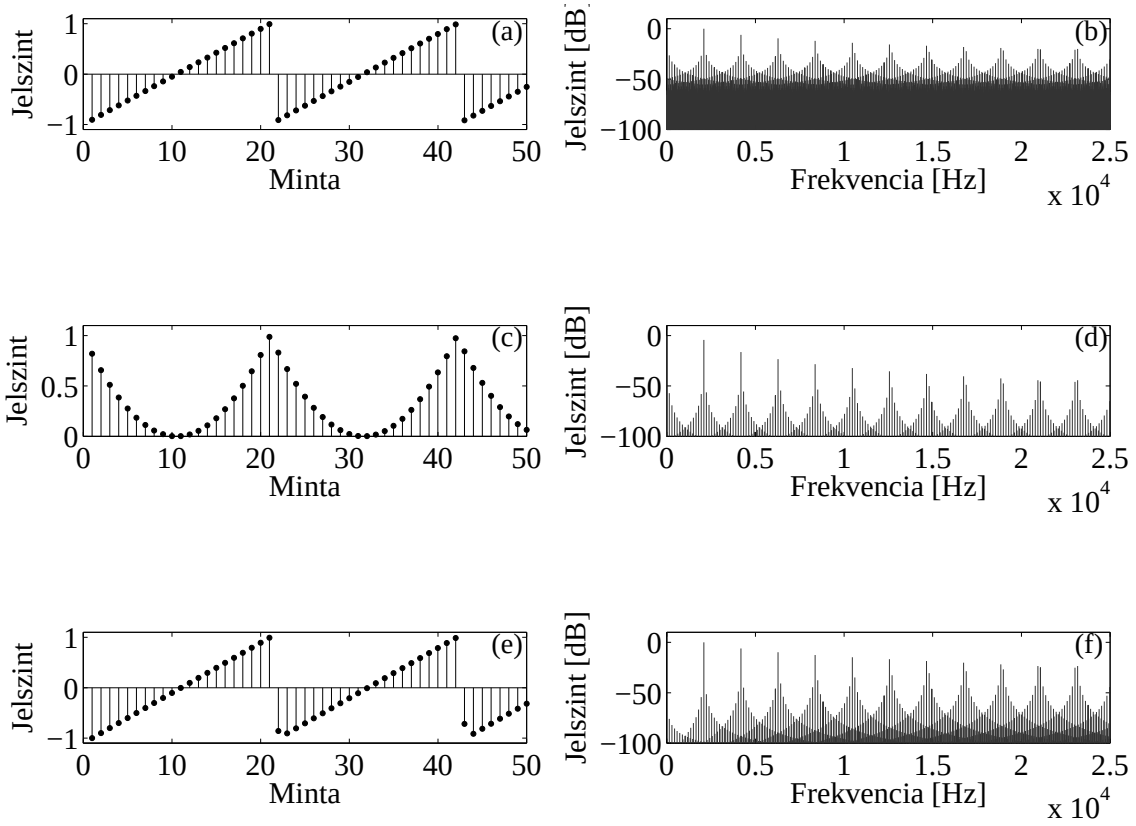
A Differentiated Polynomial Waveform (DPW) algoritmusnak először az elsőfokú verziója jelent meg, mely Differentiated Parabolic Waveform [2] névre hallgatott. A

módszer alapgondolata az, hogy a triviális megoldással ellentétben nem szimplán az analóg jelet mintavételezzük, hanem annak az integrálját. Analóg fűrészjel esetén a spektrum eredeti 6 dB-es dekádonkénti csökkenése az integrálással a duplájára nő. Így ekkor mintavételezve a jelet az átlapolódás sokkal kisebb mértékű, mint a triviális esetben. Hogy a kívánt hullámformát visszaállítsuk, a jelet diszkrét időben differenciáljuk, végül pedig skálázzuk.

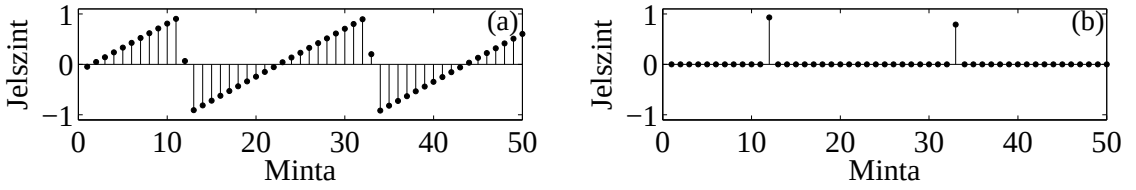
A fűrészjel időfüggvénye szakaszonként $x(t)$. A folytonos időben integrált jel így szakaszonként az $x^2/2$ kifejezéssel írható le, és ezt a jelet kell mintavételezni. A gyakorlatban tehát ezt a diszkrét $x^2(n)$ jelet generáljuk első lépésként, amit egy alap $x(n)$ fűrészjel négyzetre emelésével tehetünk meg. (A folytonos integrál 1/2 szorzójával most nem kell törődni, hiszen a generálás egy skálázással végződik.) Az $x(n)$ jel -1 és 1 között fut, és mintánként $2f_s/f$ értékkel növekszik, ahol f_s a mintavételi frekvencia és f az alapharmonikus frekvenciája. A 2.1. ábrán megfigyelhető a lépések hatása mind idő-, mind frekvenciatartományban. A minták négyzetre emelését követően deriválunk, ami az $1 - z^{-1}$ átvitelű szűrőnek felel meg. Két egymásutáni érték különbsége alacsony, ezért ahhoz, hogy a végső jel is -1 és 1 között mozogjon, skálázásra van szükség. A skálázó együttható értéke $c = f_s/(4f)$. A 2.1. (b) és (f) ábrákat összehasonlítva látható, hogy az átlapolódás jelentősen csökken. A 2.2. ábrán megfigyelhető, hogy a kapott hullámforma mindössze egy mintában tér el az alapjeltől, ez a különbség a töréspont helyén található.

Az irodalom [10] nem csak fűrészjel, hanem egyéb egyszerűbb jelek előállítására is alkalmazza az algoritmust, köztük a háromszögjelre is. Azonban csak a szimmetrikus esetről esik szó, a módszer kiterjesztése az aszimmetrikus esetre a szakdolgozatomban [11] található. A jelgenerálás menete $N = 2$ esetre a 2.3. ábra segítségével követhető.

Kezdetben adott egy aszimmetrikus triviálisan előállított háromszögjel (2.3(a) ábra). Legyen a növekvő szakasz meredeksége $A > 0$, ekkor a csökkenő szakaszé $B = -A/(A-1) < 0$. A [10] forrásban foglaltaknak megfelelően először egy $x^2 - 1$ polinomfüggvény hat az alapjelre, melynek eredménye a 2.3(b) hullámforma. Ezután meg kell szorozni egy négyszögjellel, melynek szimmetriája megegyezik a háromszögjel szimmetriájával, így a hullámforma parabolikus szakaszai felváltva pozitívak és negatívak lesznek. A szimmetrikus háromszögjel esetén a négyszögjel generálását az alapjel generálásával együtt a számlálási irány alapján lehet elvégezni, a háromszögjel növekvő szakaszán a négyszögjel értéke $+1$, csökkenő szakaszán pedig -1 [10]. Azonban ennek a 1 amplitúdójú négyszögjelnek a használatával a parabolikus szakaszok csúcsa is 1 abszolút értékű, viszont a szakaszok szélessége különböző. Ennek eredménye az lenne, hogy a szakaszok határán hirtelen változik a meredekség, ami a deriválás miatt a végleges háromszögjelben ugrásokat eredményezne. Ennek a problémának a kiküszöbölése egyszerű, és ez a lépés az egyetlen igazi különbség

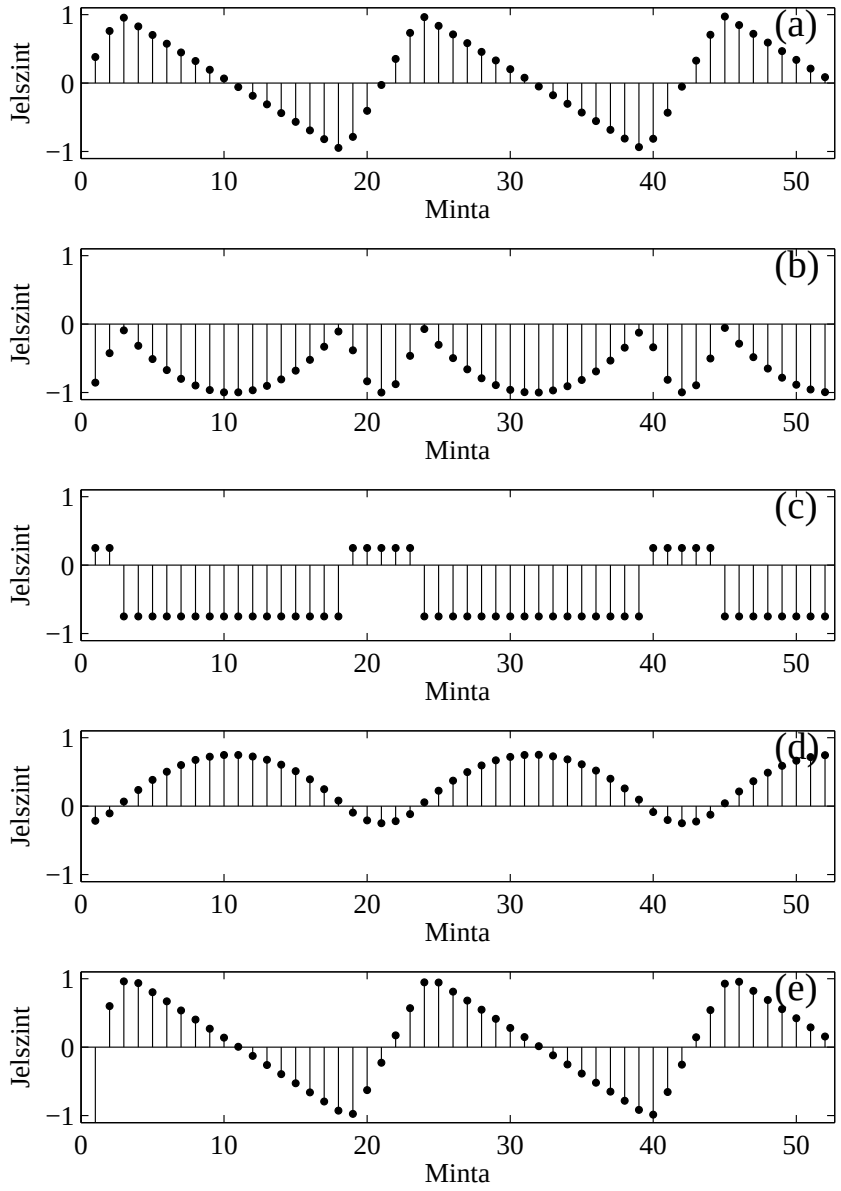


2.1. ábra. Fűrészjel generálása DPW algoritmussal: (a) triviális alapjel és (b) spektruma, (c) a négyzetre emelés utáni jel és (d) spektruma, (e) a deriválás és skálázás utáni végső jel és (f) spektruma



2.2. ábra. (a) DPW algoritmussal generált fűrsézjel és (b) a triviális hullámformától való eltérése

a szimmetrikus és az aszimmetrikus háromszögjel előállítása között. A négyszögjel megfelelő szakaszait $1/|A|$ és $1/|B|$ értékekkel kell skálázni, így az átmeneteknél is folytonosan deriválható lesz a hullámforma (lsd. 2.3. ábra (c) és (d)). Így a lépéseket egybevonva a jelformáló függvény a növekvő szakaszon $(x^2-1)/A$, a csökkenő szakaszon pedig $(x^2-1)/B$ (A ± 1 szorzó az A és B értékekben található). Végül egy deriválás és megfelelő skálázás után előáll a kívánt aszimmetrikus háromszögjel (2.3(e) ábra).



2.3. ábra. Változtatható szimetriájú háromszögjel generálása a DPW algoritmussal. A triviális jelet (a) először átadjuk a $x^2 - 1$ függvénynek. Az eredményt (b) ezután megszorozzuk (c) egy skálázott négyszögjellel, hogy megkapjuk a (d) parabolikus jelet. Végül deriválás és skálázás után kapható meg a (e) kívánt sávkorlátozott hullámforma.

Magasabbfokú eset

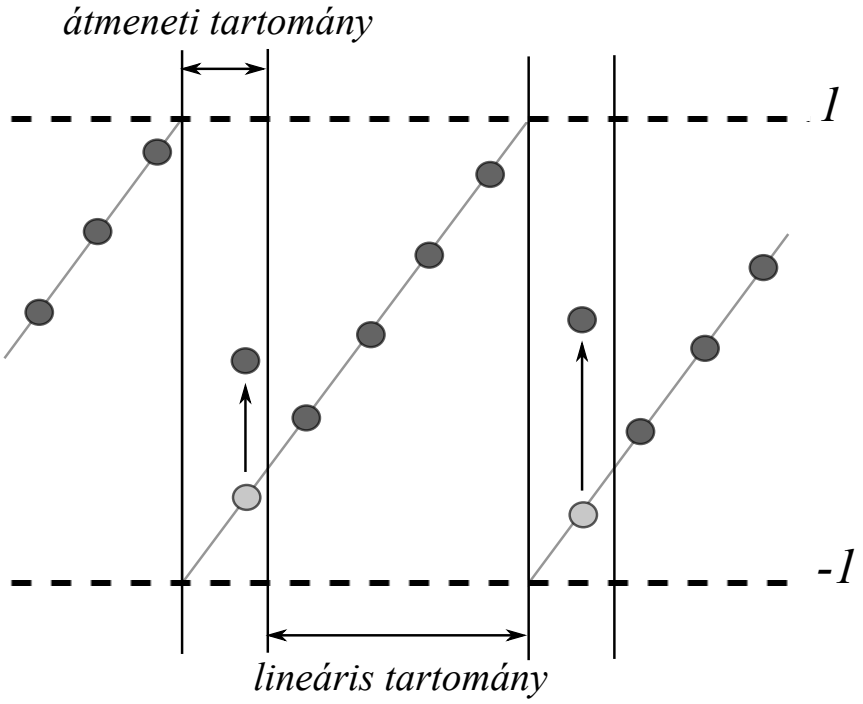
A Differentiated Polynomial Waveform módszer ugyanezt az elvet vitte tovább magasabb fokszámokra [12]. A triviális alapjelet nem második hatványra emeljük, hanem egy (N-hez tartozó adott) N-edfokú polinomfüggvényét vesszük. Fűrészjel esetén a spektrum meredeksége $6N$ dB/dekád meredekséggel csökken, mintavételezéskor így nagyobb fokszámok esetén még kisebb az átlapolódás. Végül diszkrét időben el kell végezni $N - 1$ differenciálást, hogy a végső jel előálljon. Kisebb átla-

polódással rendelkező jel előállításához tehát magasabb fokszám szükséges, azonban ez természetesen növeli az algoritmus bonyolultságát és a számításigényét.

A generált hullámforma mindössze $N-1$ pontban különbözik a triviális jeltől, mégpedig a töréspont környékén. A DPW algoritmus azonban minden egyes mintán elvégzi a polinomfüggvény szerinti számítást és differenciálást. A számítások hatékonyságát kívánta javítani a következő algoritmus.

2.3.2. A Polynomial Transition Regions algoritmus

A Polynomial Transition Regions módszer a DPW algoritmus lépéseit használja fel, így a generált jel megegyezik azzal, amit a DPW állít elő. Ehhez a minták értékét zárt alakban adja meg [13]. A jelben két tartományt különböztet meg, a lineáris szakaszt és az átmeneti szakaszt, ezek láthatóak a 2.4. ábrán. Az átmeneti tartományban találhatóak azok a minták, amelyek az alapjelhez képest megváltoznak, így az átmeneti szakasz az N-edfokú algoritmus esetén $N-1$ minta széles lesz. A maradék minta a lineáris részben található meg.



2.4. ábra. A lineáris és átmeneti tartományok szemléltetése a PTR algoritmus elsőfokú esetére.

$N=2$ esetén a fűrészjel átmeneti tartományában tehát egy minta található. A jel $y(n)$ időfüggvénye az alábbi alakban írható fel:

$$y(n) = \begin{cases} y_A(n) & \text{ha } p(n) < -1+2S \\ y_B(n) & \text{ha } p(n) \geq -1+2S, \end{cases} \tag{2.1}$$

ahol $y_A(n)$ az átmeneti, $y_B(n)$ a lineáris tartományban található jel értéke, $p(n)$ a triviális fűrészjel és $S = f/f_s$. A jel mintánként $2f/f_s = 2S$ értékkel növekszik, így valóban csak egy minta kerül be az átmeneti tartományba. Ez a minta a szakadást követő első minta.

Először következzen a lineáris szakaszban lévő minták levezetése, melyet a DPW algoritmus lépesi alapján végzünk el. Legyen az alapjel egy mintája $p(n) = p_n$ értékű. Ekkor a fűrészjel jellege miatt az előző minta $p_{n-1} = p_n - 2S$. A két mintát négyzetre kell emelni, majd diszkrét időben deriválni, azaz a két minta különbségét venni. Végül a skálázás ($c = f_s/(4f) = 1/(4S)$) a következő kifejezést adja:

$$y_B(n) = \frac{p_n^2 - p_{n-1}^2}{4S} = p_n - S. \tag{2.2}$$

Ebben a szakaszban tehát az alapjel lényegében valóban változatlan marad, mindössze egy offset a különbség. Ez csak egy összeadás műveletet igényel, így a számításigény ebben a szakaszban jelentősen csökkent.

A levezetés menete ugyanígy történik az átmeneti tartományban is. Amikor a $p(n)$ alapjel egy mintája $+1$ felett lenne, lejjebb ugrik 2 értékkel. Ez a p_n minta lesz az átmeneti tartományban. Az ugrás miatt $p_{n-1} = p_n + 2 - 2S$. Elvégezve a DPW módszer lépeseit:

$$y_B(n) = \frac{p_n^2 - p_{n-1}^2}{4S} = \frac{p_n^2 - (p_n + 2 - 2S)^2}{4S} = \left(1 - \frac{1}{S}\right) p_n - \frac{1}{S} - S + 2. \tag{2.3}$$

Ebben a tartományban a módosított minta értéke az alapjel mintájának elsőfokú függvénye. A szükséges műveletek egy szorzás és egy összeadás, tehát valóban gyorsabb elvégezni a számításokat ebben a szakaszban is, mint a DPW algoritmussal.

A következő kódrészlet a PTR algoritmust valósítja meg:

```
p = p + S;
if p > 1
  p = p - 1;
s = 2*p - 1;

if p < T
  y = correct(s);
else
  y = s - S;
```

ahol correct(s) a (2.3) képletet megvalósító függvény.

A következő fejezet egy új algoritmust mutat be, amely tovább csökkenti a jelgenerálás számításigényét.

3. fejezet

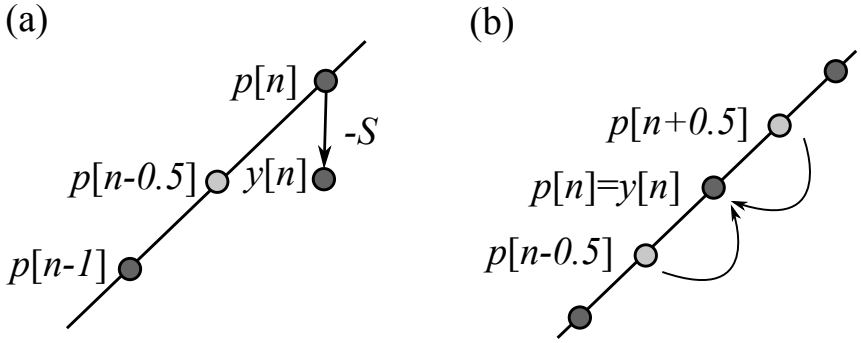
Az Efficient Polynomial Transition Regions algoritmus

3.1. Fűrészjel

A 2.3.2. fejezetben láthattuk a PTR algoritmus által használt zárt alakokat, amelyek segítségével a minták értékei kiszámolhatóak. A lineáris szakaszon ez a következő volt:

$$y_B(n) = \frac{p_n^2 - p_{n-1}^2}{4S} = p_n - S. \tag{3.1}$$

A $-S$ offset miatt a jel értéke az analóg jel annak a pontjának felel meg, amely az alapjel jelenlegi mintájához képest fél mintavételi időt késik. Ennek szemléltetése a 3.1(a). ábrán látható. Ez a késés a diszkrét idejű deriválás késleltetéséből származik. Hamarosan látni fogjuk, hogy a lineáris szakaszban a számításigény jelentősen csökkenthető, mivel ez az offset kiküszöbölhető, így nincs szükség az alapjel előállítása után további összeadás műveletre.



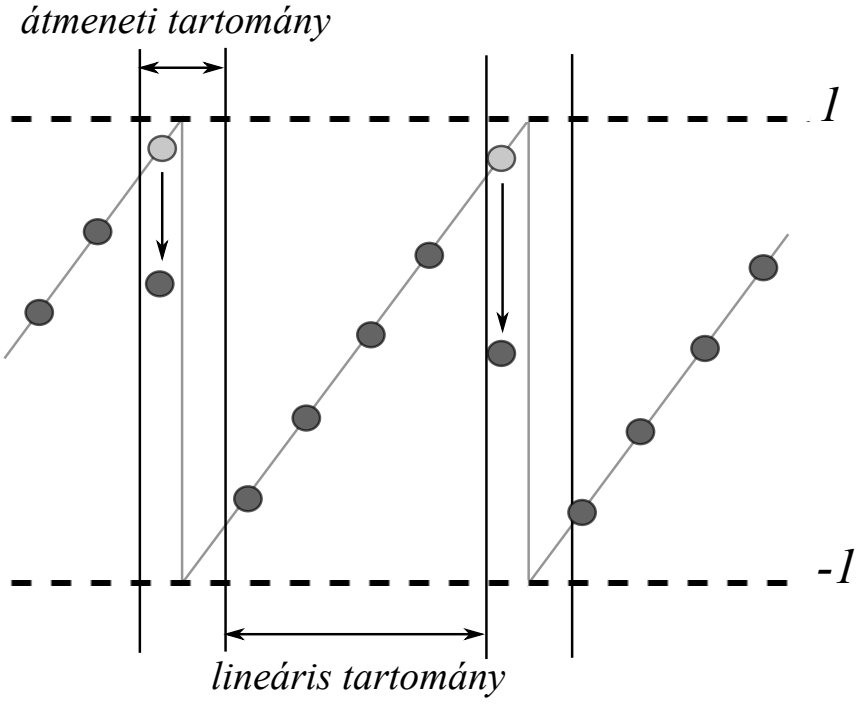
3.1. ábra. Diszkrét-idejű deriválás a (a) triviális jel (sötét pontok) és a (b) fél mintavételi idővel eltolt jel (világos pontok) segítségével.

A legegyszerűbb tehát, ha a lineáris szakaszon kimeneti jelként az alapjelet használjuk, azaz a p_n értéket. Ez megfelel annak, mintha nem az eredeti alapjelet hasz-

nálnák a jelgeneráláshoz, hanem olyat, amelyik fél mintavételi idővel siet, ahogy a 3.1(b) ábrán látható. A $p_{n-0.5} = p_n - S$ és $p_{n+0.5} = p_n + S$ tehát az analóg fűrészjel azon pontjainak felel meg, amelyek az alapjel jelenlegi értékéhez képest fél mintavételi idővel előrébb vagy hátrébb vannak. Ezekre elvégezve a DPW algoritmus műveleteit:

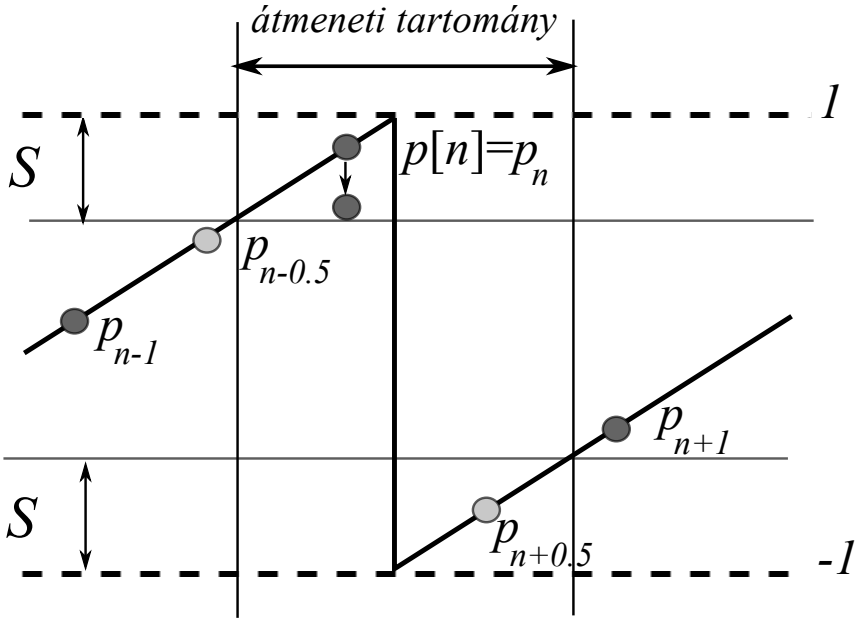
$$y(n) = \frac{p_{n+0.5}^2 - p_{n-0.5}^2}{4S} = \frac{(p_n + S)^2 - (p_n - S)^2}{4S} = p_n. \tag{3.2}$$

Tehát ugyan az eredeti triviális fűrészjelünk jeleit használjuk fel, elvben azonban a PTR algoritmust egy olyan alapjelre alkalmazzuk, amelyik fél mintavételi idővel siet. Ezt az eltolt alapjelet nem kell külön generálni, mivel az eredeti alapjel segítségével az előállítandó minták értékei zárt alakban előre megadhatóak. Az eltolás miatt azonban figyelemmel kell lenni néhány dologra. A PTR esetében az átmeneti régió a fűrészjel ugrása utáni $[0, 1]$ intervallum volt, azaz az egy módosítandó minta az ugrás és az utána egy mintavételi idő távolságban lévő pont között volt megtalálható. Az elvi eltolás hatására az átmeneti szakasz is eltolódik, így az eredeti triviális fűrészjel ugrásához képest a $[-0.5, 0.5]$ intervallumnak felel meg, ahogyan a 3.2. ábrán látható. Ez azt jelenti, hogy a módosítandó minta lehet az ugrás előtt és után is, így a két esetet (egyelőre) külön kell kezelni.



3.2. ábra. A lineáris és átmeneti tartományok szemléltetése az EPTR algoritmus elsőfokú esetére.

Nézzük először azt az esetet, amikor a módosítandó minta az ugrás előtt található ($p_n > 1 - S$). Ez az eset látható a 3.3. ábrán. Az alapjel aktuális értéke p_n, ezen érték alapján lehet eldönteni, hogy a minta az átmeneti régióban található-e. A



3.3. ábra. Az EPTR algoritmus alapgondolata. A deriválás során a triviális jel (sötét pontok) helyett az eltolt jelet (világos pontok) használjuk. A p_n értékű minta kerül módosításra.

fentieknek megfelelően tehát a fél minta távolságra lévő pontokat kell használni a PTR lépéseinek alkalmazásához. Ekkor $p_{n-0.5} = p_n - S$, a következő $p_n + S$ pedig már nagyobb +1-nél, ezért 2-vel lejjebb ugrik, így $p_{n+0.5} = p_n + S - 2$. Alkalmazva az algoritmust, a következő alak adódik:

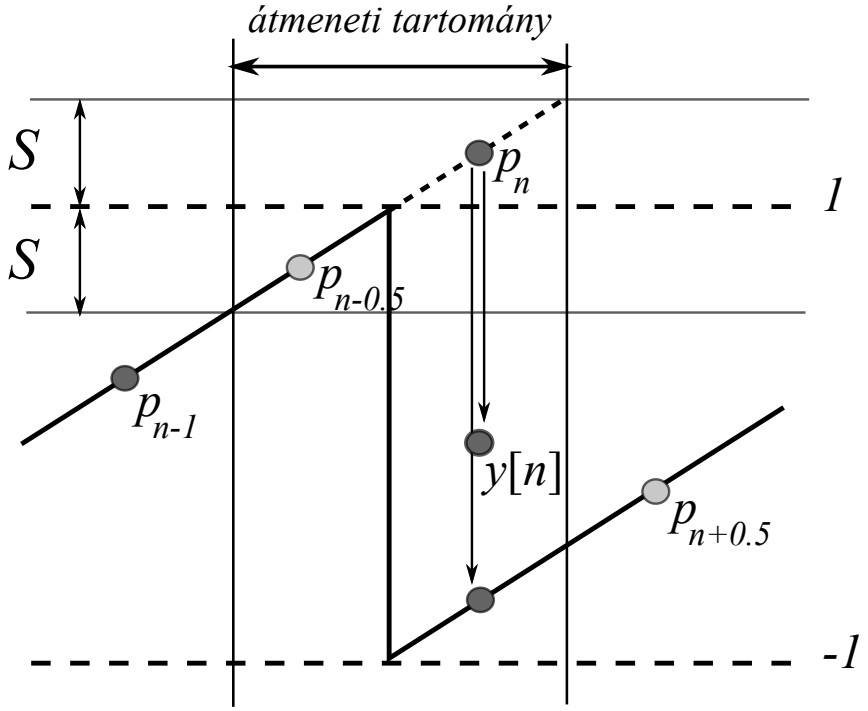
$$y(n) = \frac{p_{n+0.5}^2 - p_{n-0.5}^2}{4S} = \frac{(p_n + S - 2)^2 - (p_n - S)^2}{4S} = p_n - \frac{p_n}{S} + \frac{1}{S} - 1. \qquad (3.3)$$

A másik esetben a módosítandó minta az ugrás után található ($p_n < -1 + S$). A felhasznált pontok értékei $p_{n+0.5} = p_n + S$, valamint, mivel $p_n - S < -1$, az előző pont $p_{n-0.5} = p_n - S + 2$. Ekkor a kimeneti jel értéke:

$$y(n) = \frac{p_{n+0.5}^2 - p_{n-0.5}^2}{4S} = \frac{(p_n + S)^2 - (p_n - S + 2)^2}{4S} = p_n - \frac{p_n}{S} - \frac{1}{S} + 1. \qquad (3.4)$$

A PTR algoritmus adott triviális alapjellel dolgozik, azaz külön kezeli az alapjel és a kívánt jel előállítását. Az EPTR esetében ez bonyolultabbá tenné a feladatot, hiszen így két elágazásra lenne szükségünk: egy az alapjelben található fűrészjel megtalálásra, egy pedig annak eldöntésére, hogy a minta az átmeneti tartományban van-e. Míg az PTR esetében az ugrás megtalálása egyértelműen jelezte, hogy a következő minta a módosítandó, itt viszont ez a minta az ugrás előtt is lehet. Ez elkerülhető azzal, hogy összevonjuk a triviális jelgenerálás és a mintamódosítás lépéseit.

A módszer szemléltetése a 3.4. ábrán látható. Az összevonásnak köszönhetően az alapjel nem egy fix, -1 és $+1$ között mozgó hullámforma, hanem szükség esetén $+1$ fölé is futtathatjuk, anélkül, hogy ugrana. Így leegyszerűsödik az az eset, amikor a módosítandó minta az ugrás után található, ugyanis a triviális jel ugratása előtt kiszámolható a módosítandó jel értéke. Az ugrás után az alapjel értéke $p_n - 2$ lesz, ezt behelyettesítve (3.4)-be a (3.3) képletet kapjuk eredményül, így ugyanaz a képlet használható mindkét esetre, nincs szükség külön vizsgálatra. Amikor detektáljuk, hogy az alapjel az átmeneti tartományban található (azaz $p_n > 1 - S$), alkalmazzuk a (3.3) formulát, majd ezután ugrik az alapjel, a módosítandó mintha helyétől függetlenül.



3.4. ábra. Az EPTR algoritmus szemléltetése fűrészjel esetén. Amikor a számláló (sötét) értéke $1 - S$ fölé ér, a minta értéke módosításra kerül, majd a számláló leugrik.

A végső fűrészjel generátor algoritmus az alábbi kódrészlettel írható le:

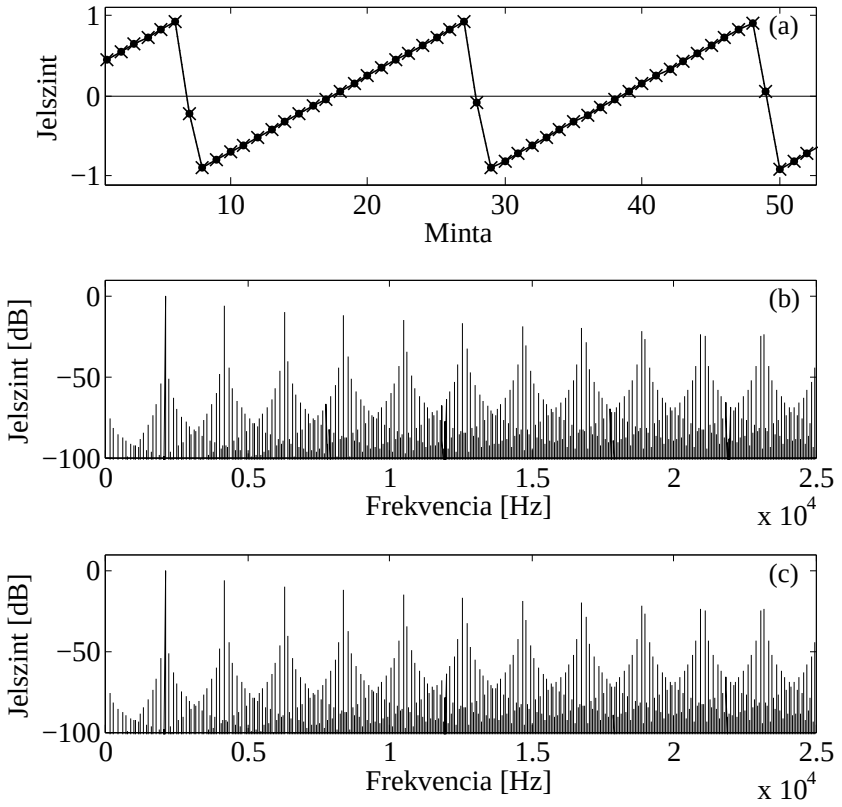
```
p = p + 2*S;
if p > 1 - S
   y = correct(p);
   p = p - 2;
else
   y = p;
```

A correct(p) függvény felelős a módosítandó minta korrigálásáért, és az eddigieknek megfelelően:

$$\text{correct}(p) = p - \frac{p}{S} + \frac{1}{S} - 1. \tag{3.5}$$

Érdemes megemlíteni, hogy az eredmény ugyanez lenne, ha az átmeneti tartomány megtalálásakor először ugrana az alapjel, és azután a (3.4) képletet alkalmaznánk.

Az algoritmus ugyanazt a jelet állítja elő, mint a PTR (és DPW), ahogyan ezt a 3.5. ábra is mutatja. A PTR algoritmushoz használt alapjel kezdőfázisát fél mintavételnyi időnek megfelelő értékkel módosítottuk, hogy a hullámformák időábrái megegyezzenek.



3.5. ábra. (a) A DPW (keresztek) és az EPTR (pontok) algoritmussal generált fűrészjelek időfüggvénye, valamint ezek spektrumai ((b) DPW, (c) EPTR).

3.2. Háromszögjel

Az aszimmetrikus háromszögjel generálása ugyanazon az elven történik, mint a fűrészjelé, azaz a számításokhoz az alapjel fél mintavételi idővel eltolt értékeit használjuk.

A háromszögjel négy szakaszból épül fel, egy növekvő és egy csökkenő lineáris szakaszból, valamint egy maximum és egy minimum csúcshoz tartozó átmeneti

tartományból. Mindkét tartomány az eddigiekhez hasonlóan $N = 2$ esetén 1 mintavételi idő széles, és így 1 mintát tartalmaz. A növekvő szakasz meredeksége $A > 0$, a csökkenő szakaszé $B = -A/(A-1) < 0$. A fűrészjel meredekségét tekintjük 1 értékűnek, az A és B értéke ennek megfelelően módosul, pl. a szimmetrikus háromszögjel esetén $A = 2$ és $B = -2$. Két egymást követő minta közötti különbség így $2AS$, illetve $2BS$. A 2.3.1. fejezetben leírtak alapján a hullámformáló függvények a növekvő szakaszon $(x^2-1)/A$, a csökkenő szakaszon pedig $(x^2-1)/B$.

Elvárható eredmény, hogy a lineáris szakaszon továbbra is az alapjel aktuális értéke legyen használható, és ez így is van:

$$y(n) = \frac{(p_{n+0.5}^2 - 1)/A - (p_{n-0.5}^2 - 1)/A}{4S} = \frac{(p_n + AS)^2 - (p_n - AS)^2}{4AS} = p_n. \quad (3.6)$$

A (3.6) levezetés a növekvő szakaszhoz tartozik, természetesen a csökkenő szakaszra is ugyanez az eredmény adódik.

Az átmeneti tartományok már bonyolultabb számítást igényelnek. Az algoritmust a 3.6. ábra szemlélteti. Következzen először a maximum csúcs $[-0.5, 0.5]$ környezetében található minta értékének meghatározása. (A minimum csúcs esete természetesen hasonló lesz.) Az alap háromszögjel A meredekséggel $+1$-ig nő, majd B meredekséggel csökken. Amennyiben a törés előtti utolsó minta értéke p_n, a törés utáni első minta a csökkenő szakaszon $p_{n+1} = 1 + (p_n + 2AS - 1)\frac{B}{A} = 1 + 2BS + (p_n - 1)\frac{B}{A}$ értékű. B negatív, így helyesen 1 alatti szám lesz.

A számoláshoz tehát szükség van az aktuális mintától fél mintavételi idő távolságra lévő értékekre. Amennyiben a módosítandó minta a törés előtt van ($p_n > 1 - AS$), és az alapjel jelenlegi értéke p_n, az előtte lévő használt érték $p_{n-0.5} = p_n - AS$. A következő már nagyobb lenne $+1$-nél, így már a csökkenő szakaszon található, értéke $p_{n+0.5} = p_n + BS + (p_n - 1)\frac{B}{A}$, hiszen p_{n+1}-hez képest egy fél mintavételi idővel van hátrébb, azaz $|BS|$ értékkel nagyobb. Ekkor alkalmazva a DPW algoritmus lépéseit:

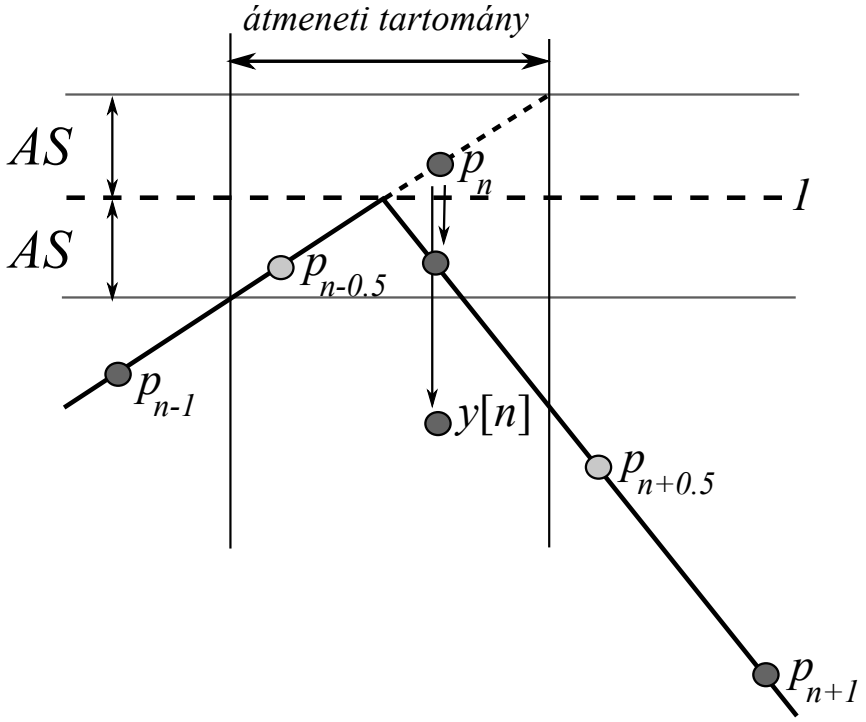
$$y(n) = \frac{(p_{n+0.5}^2 - 1)/B - (p_{n-0.5}^2 - 1)/A}{4S} = a_2 p_n^2 + a_1 p_n + a_0, \quad (3.7)$$

ahol

$$a_2 = \frac{-1}{4(A-1)S}, \quad a_1 = \frac{2AS - 4S + 2}{4(A-1)S}, \quad a_0 = \frac{-(AS-1)^2}{4(A-1)S}. \quad (3.8)$$

Hasonló a számolás menete, amikor az átmeneti tartományhoz tartozó minta a törés után helyezkedik el, azaz $p_n < 1 + BS$. Ekkor $p_{n+0.5} = p_n + BS$, valamint $p_{n-0.5} = 1 + AS + (p_n - BS - 1)\frac{A}{B}$, majd az $y(n)$ értékét a megszokott módon kell számolni. Azonban alkalmazhatjuk a 3.1. fejezetben látott trükköt, miszerint a jelgenerálás és korrekció összevonásával a számlálót $+1$ fölé is futtathatjuk, így a csúcs közelében is egy képlet, a (3.7) formula adja meg mindkét esetben a végleges

minta értékét.



3.6. ábra. Az EPTR algoritmus szemléltetése háromszögjel esetén. Amikor a számláló (sötét) értéke $1 - AS$ fölé ér, a minta értéke módosításra kerül, majd a számláló leugrik a csökkenő szakaszra. A módszer hasonló a minimum csúcs közelében is.

A minimum csúcs közelében hasonló az eljárás. Amikor a csökkenő szakaszon lévő utolsó minta p_n, a növekvő szakaszon az első értéke $p_{n+1} = -1 + 2AS + (-1 - p_n)\frac{A}{B}$. Ezek alapján már könnyedén meghatározhatóak a $p_{n-0.5}$ ás $p_{n+0.5}$ kifejezések, és eredményül hasonlóan egy képlet, a (3.9) használható minden esetre.

$$y(n) = \frac{(p_{n+0.5}^2 - 1)/A - (p_{n-0.5}^2 - 1)/B}{4S} = b_2 p_n^2 + b_1 p_n + b_0, \tag{3.9}$$

ahol

$$b_2 = \frac{-1}{4(B+1)S}, \quad b_1 = \frac{2BS + 4S - 2}{4(B+1)S}, \quad b_0 = \frac{-(BS+1)^2}{4(B+1)S}. \tag{3.10}$$

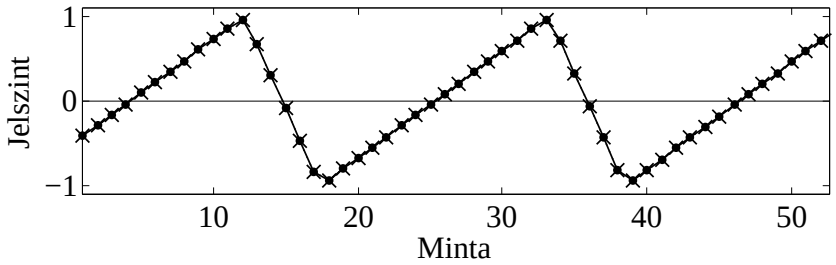
Az algoritmus tehát nagyon hasonló a fűrészjelnél látottakhoz. Egy számlálót futtatunk a háromszögjel alakjának megfelelően, amelynek aktuális értéke alapján dől el, hogy az átmeneti tartományban található-e a minta. Ha nem, a számláló értéke kerül a kimenetre, ha pedig igen, egy megfelelő függvény korrigálja a minta értékét, majd a számláló ugrik és a számlálási irány is változik. Az alábbi kód ezt az algoritmust valósítja meg.

```
if dir == 1     // felfele szamol?
  p = p + 2*A*S;
```

```
    if p > 1 - A*S    //atmeneti tartomany?
      y = correctMax(p);
      p = 1 + (p - 1)*B/A;
      dir = -1;
    else   //linearis tartomany
      y = p;
  else    // lefele szamol
    p = p + 2*B*S;
    if p < -1 - B*S   //atmeneti tartomany?
      y = correctMin(p);
      p = -1 + (p + 1)*A/B;
      dir = 1;
    else   //linearis tartomany
      y = p;
```

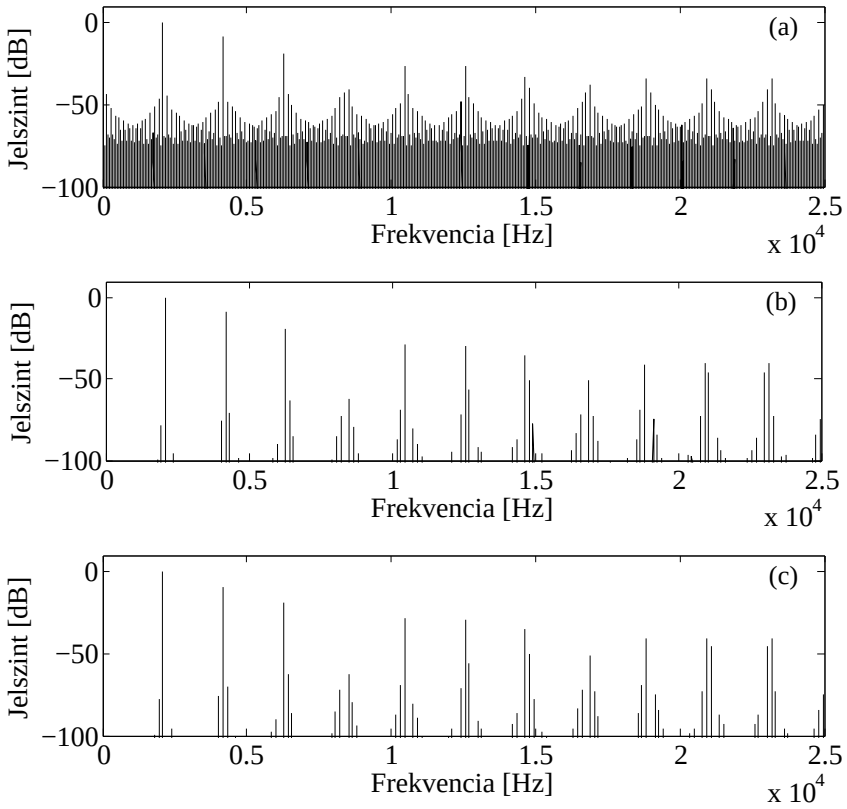
A correctMax(p) függvény a (3.7), a correctMin(p) pedig a (3.9) képletnek felel meg.

Ahogyan az a 3.7. és 3.8. ábrákon megfigyelhető, az előállított jel megegyezik a DPW és PTR algoritmussal generált jelekkel.



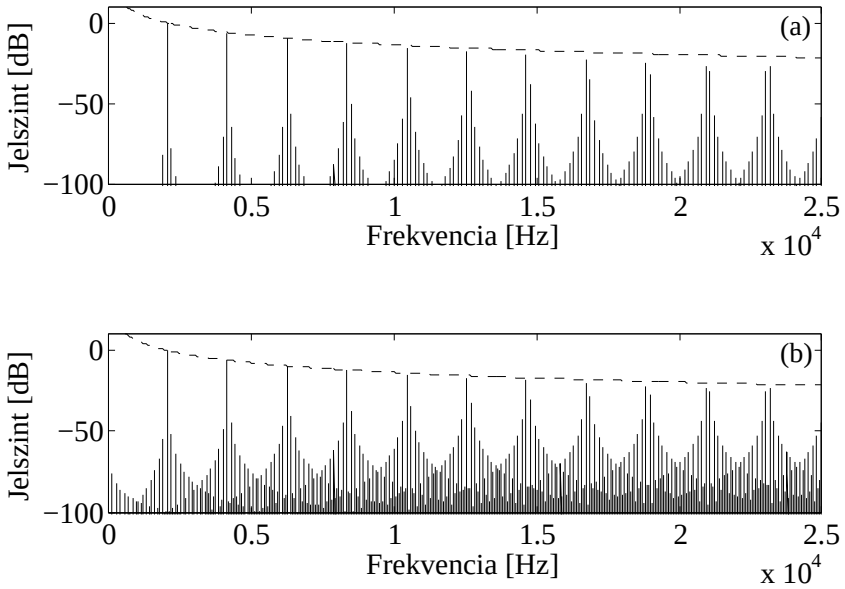
3.7. ábra. A DPW (keresztek) és az EPTR (pontok) algoritmussal generált háromszögjelek időfüggvénye. A DPW algoritmushoz használt alapjel kezdőfázisát fél mintavételnyi időnek megfelelő értékkel módosítottuk, hogy a hullámformák között ne legyen fáziskésés.

A fenti algoritmus feltételezte, hogy a háromszögjel egyik lineáris szakaszának sem olyan nagy a meredeksége, hogy a szakasz beférjen két minta közé. Ennek feltétele, hogy $|A| \leq 1/S = f_s/f$. Ugyan ennél nagyobb meredekségek esetén is megvalósítható az algoritmus, azonban a bonyolultsága jelentősen megnő. Sok különböző esetet kéne megvizsgálni arra nézve, hogy hol található a korrigálandó minta, ugyanis lehet a nagy meredekségű szakasz előtt, után vagy éppen belül. Ez akkor jelenthet problémát, amikor a szimmetria modulálása során a határesetig akarunk eljutni, azaz a háromszögjelből folytonosan áttérünk fűrészjelre. Azonban a legnagyobb megengedett meredekségű háromszögjel is már elég jól közelíti a fűrészjelet, így a további számítások bevezetése nem feltétlenül kifizetődő. A 3.9. (a) ábrán



3.8. ábra. (a) A triviális, (b) a DPW és(c) az EPTR algoritmussal előállított 25%-os aszimmetrikus háromszögjelek spektrumai.

látható a végtelen meredekséggel rendelkező fűrészjel, a (b) ábrán a megengedett háromszögjel spektruma látható. Utóbbi használatának egyetlen hátránya, hogy magas frekvenciákon az amplitúdó kis csökkenése tapasztalható. Az átlapolódás viszont még kisebb, elvégre az "ugrás" környékén két minta is korrigálásra kerül, ami így már a DPW algoritmussal $N = 3$ esetben előállított fűrészjelet közelíti.



3.9. ábra. (a) A legnagyobb megengedett meredekségű háromszögjel és (b) a fűrészjel spektruma. A szaggatott vonal az ideális spektrum burkológörbéje.

4. fejezet

Összehasonlítás

Az Efficient Polynomial Transition Regions ugyanazt a jelet állítja elő, mint a DPW és a PTR algoritmusok. Különbség a számításigényben vehető észre, így ebben a fejezetben ezt vizsgáljuk. A DPW és a PTR módszerek két változatban szerepelnek. Az "eredeti" módszer a cikkekben [2, 13] és a mellékelt kódban [14] látottakat követi, kivéve a PTR módszerrel generált háromszögjel esetében, mivel a forrásban [13] csak a fűrészjelről volt szó. Az "eredeti" PTR háromszögjel levezetése a fűrészjel alapján a szakdolgozatomban található [11]. Az "optimalizált" kódokban néhány olyan egyszerűsítés található, amik igazságosabb összehasonlítást eredményeznek. A DPW és a PTR esetében is kicseréltem az eredeti, 0 és 1 között számoló alapjelgenerátort az EPTR-ben is használt -1 és 1 között számlálóra, így megspórolva az átskálázáshoz szükséges műveleteket. Továbbá összevontam a PTR algoritmusban az alapjelgenerálást és a mintamódosítást (ahogyan az EPTR-ben is), így a két elágazás helyett csak egyre van szükség, ez pedig jelentősen gyorsította a futást.

4.1. Műveletigény

A 4.1. táblázat tartalmazza a különböző jelek esetén az egy minta előállításához szükséges műveletek átlagos számát. Az EPTR mindegyik műveletből a legkevesebbet igényli.

A triviális jelnek megfelelő számláló növeléséhez egy összeadás művelet szükséges. Ezután fűrészjel esetén az optimalizált PTR algoritmus egy elágazás során eldönti, hogy a minta az átmeneti vagy a lineáris régióban található. A lineáris régióban egy összeadás, az átmeneti szakaszon pedig egy szorzás és egy összeadás művelet szükséges. Az EPTR algoritmus eltolt alapjele megszünteti a lineáris régióban az összegzést. A többi művelet száma megegyezik. Az S értéke kicsi, a 4186 Hz frekvenciájú C_8 hang esetében is $0,1$-nél kisebb. Így az EPTR algoritmussal ilyen magas frekvenciájú hang előállításához is átlagosan $1,2$ összeadásra van szükség mintánként, míg a PTR módszer $2,1$-et igényel, azaz minden használt frekvencián

Fűrész	Összeadás	Szorzás	Elágazás
DPW (eredeti)	$3+S$	3	1
DPW (optimalizált)	$2+S$	2	1
PTR (eredeti)	$3+S$	$1+S$	2
PTR (optimalizált)	$2+S$	S	1
EPTR	$1+2S$	S	1
Háromszög	Összeadás	Szorzás	Elágazás
DPW (eredeti)	$4+2S$	4	2
DPW (optimalizált)	$3+2S$	3	2
PTR (eredeti)	$3+2S$	$1+S$	3
PTR (optimalizált)	$2+4S$	$6S$	2
EPTR	$1+4S$	$6S$	2

4.1. táblázat. A DPW, PTR and EPTR algoritmusok műveletigénye fűrész és háromszögjel esetén ($S = f/f_s$).

az EPTR bizonyul előnyösebbnek. Háromszögjel előállításakor hasonlóan a lineáris szakaszban megszűnik az összeadás művelet szükségessége, a kicsi S miatt az alkalmazott frekvenciákon mindig kevesebb az összeadások száma. Meg kell jegyezni azonban, hogy ez csak folyamatosan változó jeleknél jelent előnyt. Szakaszonként konstans hullámformák, mint pl. négyszögjel esetében az offset a PTR algoritmus használatával sem jelenik meg, így az EPTR ugyanannyi műveletet használ, mint a PTR.

4.2. Alkalmazási példa

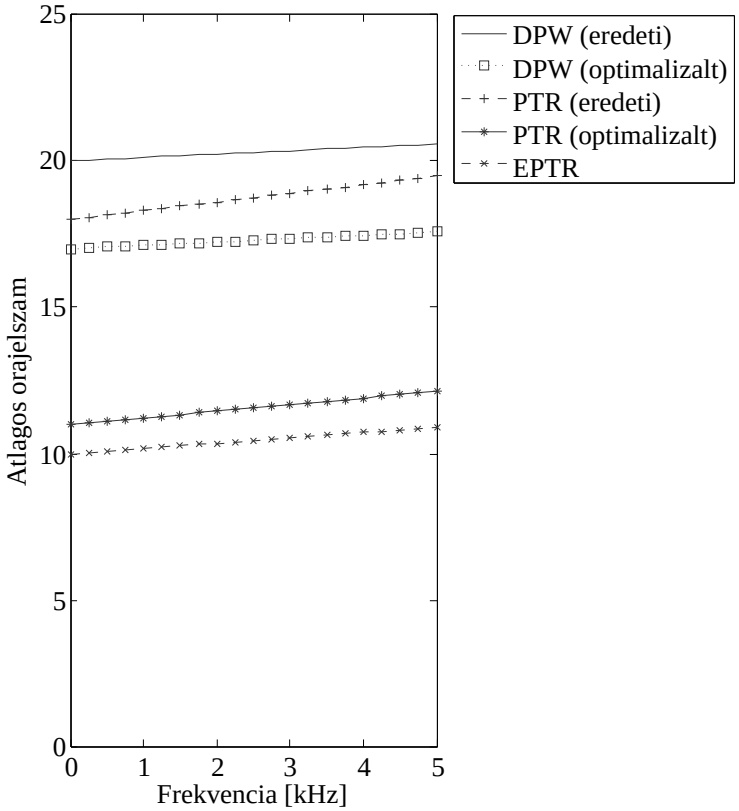
Az egyes algoritmusok számításigényét egy konkrét jelfeldolgozó processzoron is kipróbáltam. Ahhoz, hogy átfogóbb képet kapjunk az egyes módszerek számításigényéről, természetesen több különböző processzoron történő mérésre lenne szükség, ugyanakkor már egy mérés is szemlélteti az algoritmusok közötti különbségeket. A méréshez egy Analog Devices gyártmányú, ADSP-21364 Sharc típusú, 32 bites, lebegőpontos DSP-n valósítottam meg a fűrészjel-generáló algoritmusokat. A programozáshoz ADSP-21364 EZ Kit Lite típusú fejlesztőkártyát és a VisualDSP++ fejlesztőkörnyezet 4.5 verzióját használtam. A generált hullámformákat a kimeneten keresztül oszcilloszkópon lehetett ellenőrizni. Minden egyes minta kiszámítása előtt és után lekérdeztem a processzor egy hardver számlálójának értékét, így (egy offset levonása után) órajel pontossággal meg lehetett mondani a számítások idejét. A két régióban különböző műveletek hajtódnak végre, így az eltelt idő is különböző. Az egyes mintákhoz tartozó számításigényt eltároltam, így a mérés után egy periódusra nézve ki lehetett számolni az átlagos mintánként szükséges órajel periódusok számát. A felhasznált kódrészletek a függelékben találhatóak.

A 4.2. táblázat tartalmazza a mért eredményeket. A táblázat értelmezéséhez

Algoritmus	Lineáris régió	Átmeneti régió
DPW (eredeti)	20	25
DPW (optimalizált)	17	22
PTR (eredeti)	18	31
PTR (optimalizált)	11	21
EPTR	10	18

4.2. táblázat. Egy fűrészjel minta számításához szükséges órajelek száma DPW, PTR és EPTR algoritmusok használatával.

meg kell még jegyezni, hogy a DPW módszerben mintamódosítás szempontjából nem beszélhetünk lineáris és átmeneti régióról, hiszen minden mintára ugyanazok a műveletek kerülnek elvégzésre. A táblázatban az átmeneti régió azt a mintát jelenti, amelyet az alapjel ugrása után generálunk.



4.1. ábra. A különböző algoritmusok mintánkénti átlagos számítási ideje a frekvencia függvényében

A különböző algoritmusok mintánkénti átlagos számítási ideje a frekvencia függvényében a 4.1. ábrában látható. Az EPTR algoritmus minden frekvencián a leggyorsabbnak bizonyult. A használt DSP-n az optimalizált PTR algoritmushoz képest körülbelül 9%-kal, az eredetihez képest pedig átlagosan 44%-kal javítható a futási idő.

5. fejezet

Összefoglalás

Az analóg szintetizátorok modellezésének egyik fő területe az oszcillátor megvalósítása. A feszültségvezérelt oszcillátor egyszerű, széles spektrumú hullámformákat állított elő, mint a fűrész-, a háromszög-, és a négyszögjel. A triviális jelgenerálás, azaz az analóg jel mintavételezett reprezentációjának előállítása ebben az audió alkalmazásban nem megfelelő, mivel a nagymértékű átlapolódás kellemetlen hangzáshoz vezet. A probléma megoldása érdekében különböző algoritmusokat fejlesztettek ki, amelyekkel a jel sávszélességét korlátozzák, így a kisebb átlapolódásnak köszönhetően az előállított jel hangzása az analóg jelét közelíti. A dolgozat egy ilyen, új algoritmust mutat be.

A dolgozatban elsőként a Differentiated Parabolic Waveform (DPW) algoritmust tárgyaltuk, mivel ez szolgál a további módszerek alapjául. Az ötlet szerint nem a triviális jelet mintavételezzük, hanem annak integrálját. Az integrálással ugyanis a spektrum csökkenésének meredeksége megnő, ekkor mintavételezve a jelet tehát kisebbek az átlapolódott komponensek. A kívánt hullámforma létrehozásához az így kapott jelet diszkrét időben deriválni és skálázni kell. A módszer magasabb fokú kiterjesztése a Differentiated Polynomial Waveform algoritmus, melyben N-edfokú esetben $N-1$ integrálást és diszkrét deriválást kell elvégezni.

A Polynomial Transition Regions (PTR) módszer csökkenti a DPW algoritmus számításigényét. A jel egy periódusát két tartományra osztja. A lineáris szakaszban azok a minták találhatóak, amelyek (egy offsettől eltekintve) nem változnak, az átmeneti tartományban pedig a módosítandó minták. Mindkét tartományban a DPW lépéseit követve analitikusan kiszámolva, zárt alakban megadható a minták értéke az alapjel aktuális értékének függvényében. A műveletek száma a DPW-hez képest csökkent, többek között annak köszönhetően, hogy a lineáris szakaszban mindössze az alapjel és az offset összegzésére volt szükség.

A dolgozatban az Efficient Polynomial Transition Regions (EPTR) nevű új algoritmust mutattam be. A PTR-ben látott offset megszüntetésével az összeadás művelet elkerülhető, így a számításigény tovább csökkenthető. A diszkrét idejű

deriválás fél mintavételi időnyi késleltetést okoz, ez okozza az offsetet az PTR algoritmussal generált jelben. Ezért az EPTR az alapjel fél mintával eltolt értékeit használja a számításokhoz, így a DPW lépéseit alkalmazva a lineáris szakasz mintáinak értéke megegyezik az alapjel aktuális értékeivel. Az átmeneti tartományban lévő mintákhoz természetesen ugyanúgy a fél mintával eltolt értékeket kell használni. A módszer további előnye, hogy a triviális jel generálását és az átmeneti tartományban lévő minták módosítását együtt kezeli. Ezzel az ötlettel a triviális számláló az alsó és felső határokon is túl futtatható szükség esetén, és ezzel az algoritmus tovább egyszerűsödik. Amikor a triviális jel generálása közben észleljük, hogy +1-nél nagyobb a minta értéke, elvégezzük a megfelelő módosítást, a számláló pedig ugrik. Ez az eljárás egy pár soros kóddal megvalósítható. A módszert kiterjesztettem a változtatható szimmetriájú háromszögjel előállítására is. A lineáris szakaszban továbbra is az alapjel használható, a két (maximum és minimum csúcshoz tartozó) átmeneti szakaszhoz pedig levezettem a korrekciós képleteket. A szimmetria modulálásával a háromszögjel folytonosan vihető át fűrészjelbe.

Az EPTR előnye a PTR-hez képest a kisebb számításigény. Az új módszer kb. 30%-kal kevesebb műveletet igényel a fűrészjel előállításához. Az algoritmusokat a gyakorlatban, egy konkrét DSP-n implementálva is kipróbáltam. Az egyes minták előállítását órajel pontossággal lehetett mérni, így ki lehetett számolni az egy minta előállításához szükséges átlagos órajelszámot. A mérés eredménye szerint az EPTR az optimalizált PTR-nél átlagosan 9%-kal, az eredeti PTR-nél 44%-kal gyorsabb. Az EPTR tehát a jelenlegi leghatékonyabb sávkorlátozott oszcillátor algoritmusnak tekinthető.

A dolgozatban tárgyalt EPTR algoritmust a 2013 augusztusi, Stockholmban megrendezett Sound and Music Computing Conference 2013 konferencián is bemutattuk [15].

Irodalomjegyzék

[1] Synthesizer. `http://en.wikipedia.org/wiki/Synthesizer`.

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

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

[4] Amar Chaudhary. Band-limited simulation of analog synthesizer modules by additive synthesis. In *Audio Engineering Society Convention 105*, September 1998.

[5] DanaC. Massie. Wavetable sampling synthesis. In *Applications of Digital Signal Processing to Audio and Acoustics*, volume 437, pages 311–341. 2002.

[6] Tim Stilson and Julius Smith. Alias-free digital synthesis of classic analog waveforms. In *in Proc. International Computer Music Conference*, pages 332–335, Hong Kong, August 1996.

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

[8] Eli Brandt. Hard sync without aliasing. In *Proc. International Computer Music Conference*, pages 365–368, Havana, Cuba, September 2001.

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

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

[11] Ambrits Dániel. Analóg szintetizátor modellezése különböző oszcillátor algoritmusokkal. Szakdolgozat, Budapesti Műszaki és Gazdaságtudományi Egyetem, 2012.

[12] Vesa Välimäki, Juhan Nam, Julius O. Smith, and Jonathan S. Abel. Alias-suppressed oscillators based on differentiated polynomial waveforms. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(4):786–798, May 2010.

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

[14] Jari Kleimola and Vesa Välimäki. Polynomial Transition Regions [Online]. November 2011. URL: http://www.acoustics.hut.fi/go/spl-ptr.

[15] Dániel Ambrits and Balázs Bank. Improved polynomial transition regions algorithm for alias-suppressed signal synthesis. In *Proceedings of the Sound and Music Computing Conference 2013*, pages 561–568, Stockholm, Sweden, August 2013.

Függelék

F.1. Kódrészletek

A különböző algoritmusokat egy Analog Devices gyártmányú, ADSP-21364 Sharc típusú, 32 bites, lebegő pontos DSP-n is megvalósítottam. A fejlesztéshez VisualDSP++ fejlesztőkörnyezetet és C++ nyelvet használtam. Az alábbi kódok egy mintát számolnak ki, valamint megmérik, hogy ehhez mennyi időre volt szükség.

Az alábbi az órajel periódusok számának méréséhez szükséges váz. A számláló két kiértékelése közötti kommentezett részbe kell az egy mintát megvalósító kódot másolni. Az egyes mintákhoz szükséges idő a runTimeBuffer nevű tömbbe került elmentésre.

```
asm volatile("nop;");
CYCLE_COUNT_START( runTimeCounter );
asm volatile("nop;");

// egy minta szamitasa

asm volatile("nop;");
CYCLE_COUNT_STOP( runTimeCounter );
asm volatile("nop;");

runTimeBuffer[runTimeBuffPtr & 0x03FF] = runTimeCounter;
if (runTimeBuffPtr<1023) runTimeBuffPtr++;
```

Az alábbi értékek frekvencia váltásakor (pl. MIDI-üzenet lekezelésekor) előre kiértékelődnek, így az egyes algoritmusok gyorsabban lefutnak.

```
float S = S_DEF;            //    S = f / Fs
float a = 1.0 - 1.0/S_DEF;
float b = 1.0 - S_DEF;
float c = 2.0*S_DEF;
float d = 0.25/S_DEF;
```

A következő kódok pedig az egyes algoritmusokat valósítják meg. Egy kódrészlet

egy mintát számol ki, és ciklikusan futnak.

DPW (eredeti)

```
p = p + T;
if (p > 1.0)
{
    p = p - 1.0;
}
s = 2*p - 1.0;
y = (s*s - st)*d;
st = s*s;
```

DPW (optimalizált)

```
p = p + c;
if (p > 1.0)
{
    p = p - 2.0;
}
y = (p*p - st)*d;
st = p*p;
```

PTR (eredeti)

```
p = p + T;
if (p > 1.0)
{
        p = p - 1.0;
}
s = 2*p - 1;
if (p < T)
{
    y = a*s + a + 1 - T;
}
else
{
    y = s - T;
}
```

PTR (optimalizált)

```
p = p + c;
if (p > 1.0)
{
    p = p - 2.0;
    y = a*p + a + 1 - T;
}
else
{
    y = p - T;
}
```

EPTR

```
p = p + c;
if (p > b)
{
    y = a*p - a;
    p = p - 2.0;
}
else
{
    y = p;
}
```