Juri Lukkarila

Analogisen äänisynteesin digitaalinen mallinnus

Sähkötekniikan korkeakoulu

Kandidaatintyö
Espoo 10.12.2014

Vastuuopettaja:

TkT Markus Turunen

Työn ohjaaja:

Prof. Vesa Välimäki

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä: Juri Lukkarila

Työn nimi: Analogisen äänisynteesin digitaalinen mallinnus

Päivämäärä: 10.12.2014 Kieli: Suomi Sivumäärä: 3+33

Koulutusohjelma: Elektroniikka ja sähkötekniikka

Vastuuopettaja: TkT Markus Turunen

Ohjaaja: Prof. Vesa Välimäki

Tämä kandidaatintyö esittelee menetelmiä analogisen äänisynteesin digitaaliseen mallintamiseen ja toteuttamiseen reaaliajassa. Useimmat analogiset syntetisaattorit käyttävät äänentuottomenetelmänään vähentävää synteesiä, missä spektriltään rikasta lähdesignaalia suodatetaan halutun äänen aikaansaamiseksi. Vähentävän synteesin digitaalinen toteuttaminen sisältää monia haasteita, ja aihepiiri on ollut akateemisen tutkimuksen kohteena etenkin viimeisen kymmenen vuoden ajan.

Analogisynteesin digitaalinen mallinnus voidaan jakaa kahteen osa-alueeseen, jotka ovat lähdesignaalien tuottaminen ja suodatinmallinnus. Tämä työ keskittyy lähdesignaalien tuottamiseen, mihin myös aihepiirin tutkimus on painottunut. Lähdesignaalien tuottamisessa merkittävin ongelma on digitaalisissa järjestelmissä esiintyvä laskostuminen, joka vaikuttaa negatiivisesti äänenlaatuun. Lähdesignaalien tuottamismenetelmät voidaan jakaa neljään kategoriaan, jotka toisistaan erottaa niiden lähestymistapa laskostumisen estämiseen.

Lähdesignaalien tuottamiseen käytettäviä oskillaattorialgoritmeja voidaan vertailla niiden laskennallisen kuorman, muistivaatimuksien sekä saavutettavan äänenlaadun suhteen. Yhtä täydellistä oskillaattorialgoritmia ei ole vielä olemassa, mutta useilla menetelmillä on mahdollista saavuttaa hyviä tuloksia. Digitaalisen mallinnuksen perimmäinen tavoite on tuottaa digitaalisesti oikeiden analogisyntetisaattoreiden kanssa identtiseltä kuulostavaa ääntä, mikä tyypillisesti vaatii analogisien piirien sisältämien epälineaarisuuksien mallintamista.

Avainsanat: audiojärjestelmät, äänenkäsittely, digitaalinen signaalinkäsittely, musiikki

Sisällysluettelo

Symbolit ja lyhenteet

Symbolit

c_{dc}	Vaihesiirtymä (*offset*)
$\delta(t)$	Yksikköimpulssifunktio
f	Taajuus [Hz]
f_N	Normalisoitu taajuus
f_s	Näytteenottotaajuus [Hz]
k	Indeksimuuttuja, $\in \mathbb{N}$
n	Indeksimuuttuja, $\in \mathbb{N}$
N	Asteluku, 1,2,3,...
$\mathbb{N}$	Luonnolliset luvut, 0,1,2,3,...
$p(n)$	Vaihesignaali
$s(n)$	Jakojäännöslaskuri
t	Aikamuuttuja [s]
T	Jaksonaika, $T = 1/f$ [s]
$u(t)$	Yksikköaskelfunktio
$x(t)$	Jatkuva signaali
W	Siirtymäalueen leveys

Lyhenteet

BLIT	Kaistarajoitettu impulssijono (*bandlimited impulse train*)
BLEP	Kaistarajoitettu askelfunktio (*bandlimited step function*)
DPW	Differentioitu polynominen aaltomuoto (*differentiated polynomial waveform*)
DSF	Diskreetti summauskaava (*discrete summation formula*)
EPTR	Tehokas PTR (*efficient PTR*)
FFT	Nopea Fourier-muunnos (*fast Fourier transform*)
FIR	Äärellinen impulssivaste (*finite impulse response*)
FM	Taajuusmodulaatio (*frequency modulation*)
PTR	Polynomiset siirtymäalueet (*polynomial transition regions*)

1 Johdanto

Analogiset syntetisaattorit yleistyivät muusikoiden ja musiikkituottajien keskuudessa 1960-luvulla, kun ensimmäiset kaupalliset analogiset syntetisaattorit tulivat markkinoille. 1970-luvulle tultaessa etenkin Robert Moogin suunnittelemat syntetisaattorit olivat suosittuja muusikoiden keskuudessa. Tämän aikakauden syntetisaattorit perustuivat pääosin vähentävään äänisynteesiin (*substractive synthesis*). 1980-luvulle tultaessa uudet digitaaliset syntetisaattorit, jotka perustuivat esimerkiksi FM-synteesiin, syrjäyttivät nopeasti analogiset vastineensa. 1990-luvulla kuitenkin kiinnostus analogisiin syntetisaattoreihin ja vähentävään synteesiin heräsi jälleen etenkin elektronisen musiikin parissa. Tämä kiinnostus on jatkunut ja voimistunut aina nykypäivään saakka. [1, 2]

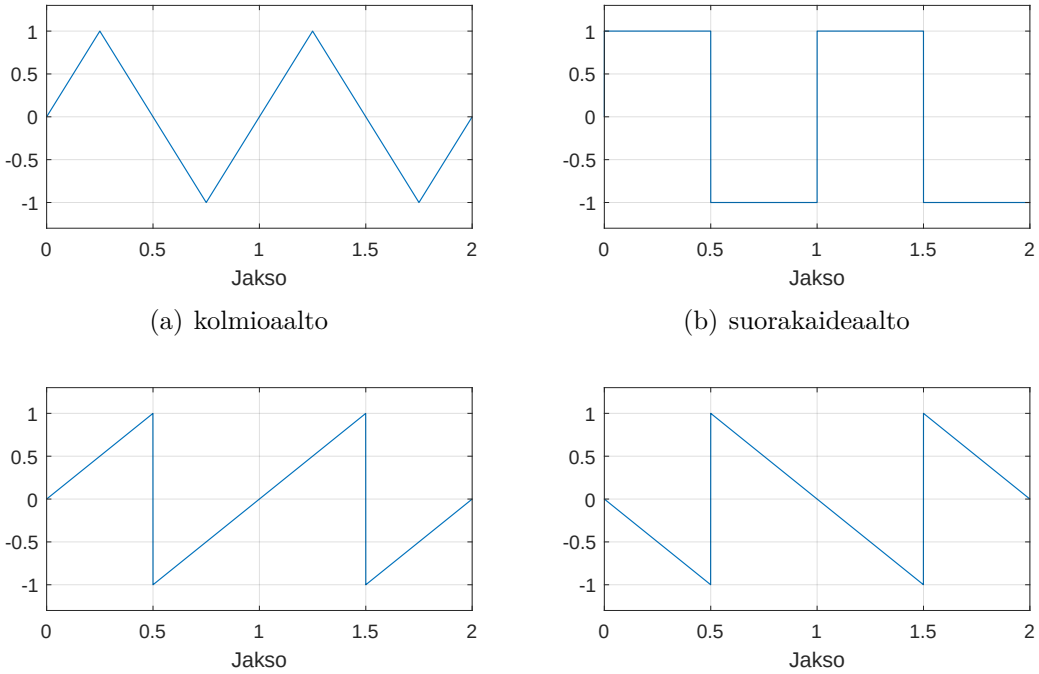
Analoginen elektroniikka asettaa kuitenkin omat haasteensa niin hinnan, käytettävyyden kuin ominaisuuksien osalta. Analogiset syntetisaattorit ovat usein kookkaita, ja niissä esiintyy viritysongelmia. Digitaalitekniikka on puolestaan halpaa ja tehokasta, ja etenkin nykyaikaiset tietokoneet suurella laskentatehollaan ja monipuolisuudellaan tarjoavat hyvin laajat mahdollisuudet äänisynteesille. Näin ollen syntyi kiinnostus mallintaa analogista vähentävää synteesiä digitaalisesti, eli jäljitellä analogisynteesin tuottamaa ääntä digitaalisilla menetelmillä ja järjestelmillä. Tästä käytetään yleisesti nimitystä virtuaalianalogisynteesi, ja siihen perustuvia kaupallisia tuotteita on ollut saatavilla aina 1990-luvun puolivälistä saakka niin fyysisinä laitteistoina kuin ohjelmistototeutuksina. [2, 3, 4]

Akateemisessa maailmassa virtuaalianalogisynteesin tutkimus on kasvanut suuresti viimeisen kymmenen vuoden aikana, ja aihetta on tutkittu paljon Aalto-yliopiston sähkötekniikan korkeakoulussa signaalinkäsittelyn ja akustiikan laitoksella. Analogisynteesin mallinnuksen lopullinen tavoite on tuottaa identtiseltä kuulostavaa ääntä mallinnuksen kohteen kanssa. Merkittävimmät haasteet tämän tavoitteen toteuttamisessa ovat digitaalisien järjestelmien rajattu kaistanleveys ja sähköisten piirien epälineaarisuuksien mallintaminen. [5]

Tämän työn tarkoitus on esitellä menetelmiä virtuaalianalogisynteesissä tarvittavien lähdesignaalien tuottamiseen. Analogisynteesin mallinnus voidaan jakaa kahteen osa-alueeseen, jotka ovat lähdesignaalien generointi ja suodatinmallinnus. Työn pääpaino on lähdesignaalien tuottamiseen käytettävissä algoritmeissa, mihin myös aiheen tutkimus on pitkälti keskittynyt. [5]

2 Vähentävä synteesi

Vähentävä synteesi on analogisten syntetisaattoreiden pääasiallinen äänentuottomenetelmä. Vähentävässä äänisynteesissä lähdetään liikkeelle spektriltään, eli taajuusjakaumaltaan rikkaasta lähdesignaalista, jota suodatetaan säädettävällä resonanssilla varustetulla alipäästösuodattimella. Lähdesignaalina toimii tyypillisesti yksi tai useampi perinteisistä geometrisistä aaltomuodoista, joita ovat kolmio-, saha- ja suorakaideaalto. Näiden lisäksi voidaan käyttää kohinaa, sekä edellä mainittujen aaltomuotojen muunnelmia pulssinleveyden suhteen. Mainitut aaltomuodot esitetään kuvassa 1. [2]



(a) kolmioaalto

(b) suorakaideaalto

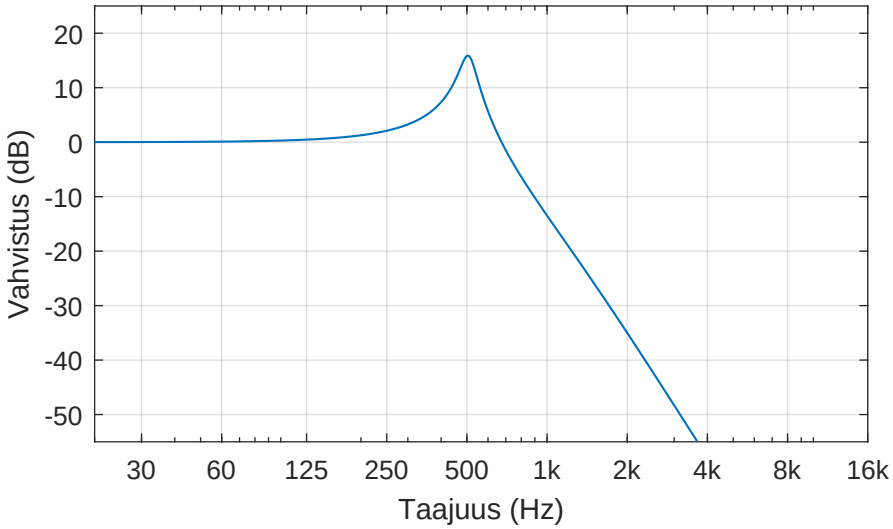
(c) saha-aalto

(d) käännetty saha-aalto

Kuva 1: *Ideaaliset geometriset aaltomuodot.*

Alipäästösuodattimella saadaan poistettua spektristä ei-halutut ylemmät harmoniset komponentit, ja vastaavasti säädettävällä suodattimen resonanssilla voidaan korostaa taajuuskomponentteja suodattimen rajataajuudella. Voimakkaan resonanssin käyttö onkin yksi vähentävällä synteesillä tuotetun äänen ominaispiirre. Kuvassa 2 esitetään tyypillinen taajuusvaste resonoivalle alipäästösuodattimelle. Vähentävän synteesin toimintaperiaate voidaan siis käsittää lähde-suodinjärjestelmänä, missä aluksi tuotetun signaalin spektriä suodattimella muokkaamaalla saadaan aikaiseksi halutunlainen lopputulos. [4]

Esiteltyjen aaltomuotojen tuottaminen analogisella elektroniikalla on helppoa ja yksinkertaista, ja näitä aaltomuotoja tuottavia erilaisia piiriratkaisuja on olemassa monia. Historiallisesti yksi merkittävä tekijä juuri näiden aaltomuotojen käyttämiseen olikin, että ensimmäisiä syntetisaattoreita suunniteltaessa näiden perinteisten aalto-



Kuva 2: *Resonoivan alipäästösuodattimen taajuusvaste.*

muotojen elektroniset toteutukset olivat jo olemassa funktiogeneraattoreiden myötä. Vähentävän synteesin digitaalisessa toteutuksessa suurimman ongelman muodostaa ideaalisten aaltomuotojen epäjatkuvuuskohdista aiheutuva kaistarajoittamattomuus, mikä tarkoittaa, että signaalin täydelliseen esittämiseen vaaditaan ääretön määrä harmonisia taajuuskomponentteja. Kaistarajoittamattomuus on selkeästi nähtävissä aaltomuotojen Fourier-sarjoista: Kolmioaallon Fourier-sarja esitetään kaavassa (1), suorakaideaallon kaavassa (2) ja saha-aallon kaavassa (3). [2, 6]

$$x_{kolmio}(t) = -\frac{4}{\pi}\sum_{k=1}^{\infty}\frac{|\mathrm{sinc}(\frac{k}{2})|}{k}\cos(2\pi k f t), \tag{1}$$

$$\text{missä } \mathrm{sinc}(x) = \tfrac{\sin(\pi x)}{\pi x}.$$

$$x_{suorakaide}(t) = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin(2\pi(2k-1)ft)}{(2k-1)} \tag{2}$$

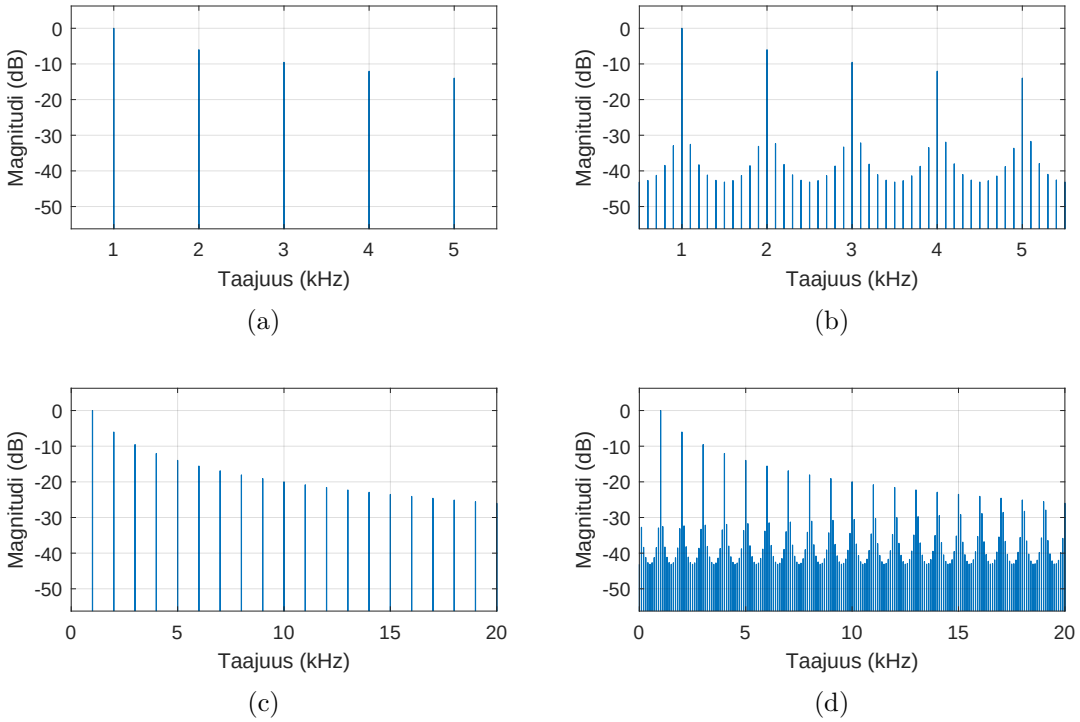
$$x_{saha}(t) = -\frac{2}{\pi}\sum_{k=1}^{\infty}\frac{\sin(2\pi k f t)}{k} \tag{3}$$

Analogisessa elektroniikassa korkeat taajuudet vaimenevat itsestään, kun taas digitaalisissa järjestelmissä kaistanleveys on tarkasti rajattu, eikä tämän rajan yläpuolella saisi esiintyä lainkaan signaalikomponentteja [2]. Rajoittuneesta kaistanleveydestä aiheutuvaa laskostumisongelmaa käsitellään osiossa 3.

3 Laskostuminen

Jatkuva-aikaisesta signaalista diskreettiin näytejonoon siirryttäessä Nyquist-Shannon-teoreeman mukaisesti diskreettiaikaisessa (digitaalisessa) järjestelmässä tarvitaan vähintään kaksinkertainen näytteenottotaajuus suurimpaan näytteistettävään signaalitaajuuteen nähden, jotta alkuperäinen signaali voidaan esittää täydellisesti [7]. Nyquist-taajuudeksi kutsutaan näytteenottotaajuuden puolikasta. Tätä taajuutta suurempitaajuiset signaalikomponentit laskostuvat, eli summautuvat takaisin Nyquist-taajuuden alapuolelle, mikä aiheuttaa vääristymiä signaaliin. Samalla Nyquist-taajuuden ylittävien komponenttien informaatio menetetään. Taajuuskomponentti, jonka taajuus on $f_s/2 + f$ laskostuu takaisin taajuudelle $f_s/2 - f$, missä f_s on näytteenottotaajuus [2]. Laskostuminen kuullaan tyypillisesti epäharmonisena särönä ja kohinamaisena häiriönä. [6]

Kuvassa 3 nähdään laskostumisen aiheuttamat ylimääräiset häiriökomponentit signaalin spektrissä, kun näytteistetään triviaalisesti tuotettua saha-aaltoa. Vasemmalla puolella on puhdas saha-aallon spektri, ja oikealla puolella laskostunut tapaus. Saha-aallon perustaajuus oli 1 kHz ja näytteenottotaajuus audiotekniikalle tyypillinen 44,1 kHz. Puhdas spektri tuotettiin laskemalla saha-aallon Fourier-sarjan ensimmäiset 20 taajuuskomponenttia, jolloin suurin taajuuskomponentti asettuu 20 kHz:n taajuudelle eikä Nyquist-taajuutta näin ollen ylitetä. Kaistarajoitetun Fourier-sarjan laskemista käsitellään tarkemmin osiossa 4.1.1.

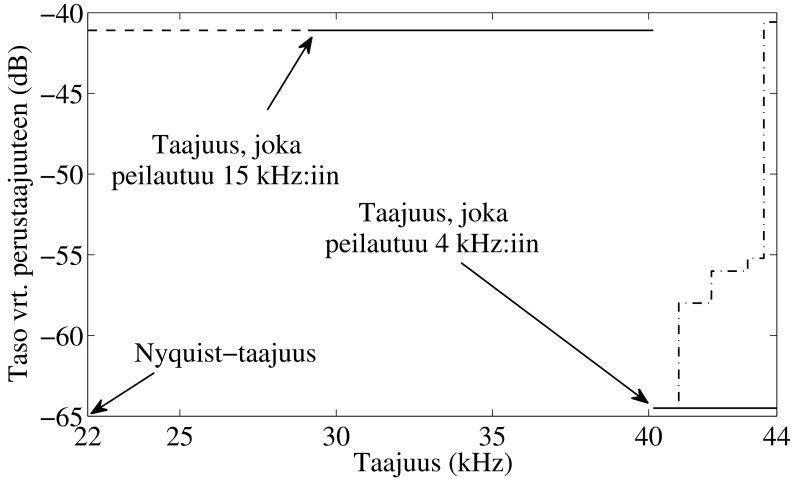


Kuva 3: *Saha-aallon spektri yhden kilohertsin perustaajuudella: puhdas (a), (c), ja laskostunut (b), (d) tapaus.*

Ihmisen kuulon psykoakustisien ominaisuuksien johdosta laskostumista voi kuitenkin esiintyä jonkin verran, ennen kuin se tulee kuultavaksi. Laskostumisen kuultavuus riippuu monista tekijöistä, kuten tuotettavan signaalin perustaajuudesta, harmonisten komponenttien määrästä ja niiden suhteellisista voimakkuuksista ja sijainneista laskostuneisiin komponentteihin nähden. Kuulon kannalta laskostumisen havaitsemiseen vaikuttaa erityisesti kaksi psykoakustiikan ilmiötä: kuulokynnyksen taajuusriippuvuus ja taajuuspeitto. [2, 6]

Kuulon herkkyys heikkenee voimakkaasti sekä pienillä että suurilla taajuuksilla, jolloin näillä taajuuksilla esiintyvät tasoltaan pienet laskostuneet komponentit jäävät kuulokynnyksen alapuolelle eli kuulematta. Taajuuspeittoilmiössä taajuusalueella lähekkäin oleva samalle kriittiselle kaistalle osuva voimakas ääni peittää itseään hiljaisemman äänen, eli suhteelliselta tasoltaan tarpeeksi paljon pienempi laskostunut komponentti peittyy likimäärin samalla taajuudella esiintyvän voimakkaamman signaalikomponentin alle. Kuulokynnyksen vaikutus painottuu perustaajuuden alapuolella, eli se rajoittaa perustaajuden alapuolelle peilautuvien häiriöiden kuuluvuutta. Käytännön tilanteissa ja oskillaattorialgoritmeja suunnittellessa riittää näin ollen, että syntyvä laskostuminen ei ole kuultavissa. [2]

Lehtonen et al. tutkivat laskostumisen kuultavuutta saha-aallossa kuuntelukokeilla. Kuuntelukokeiden perusteella saatiin minimivaatimukset laskostuneiden komponenttien vaimennukselle saha-aallon tapauksessa. Tästä saadaan edelleen saha-aallon taajuuskomponenttien suurin sallittu taso näytteenottotaajuuden yläpuolella, jolloin vielä laskostuminen pysyy kuulumattomissa. Tutkimuksen tulokset nähdään kuvassa 4. Tuloksia voidaan hyödyntää oskillaattorialgoritmien suunnittelussa, kun tiedetään kuinka paljon laskostumista milläkin taajuudella voidaan sallia. [6]



Kuva 4: *Suurimmat sallitut tasot ensimmäisen kertaluvun laskostuneille komponenteille kun $f_s = 44,1$ kHz [6, Kuva 3]. Yhtenäinen viiva kuvan oikeassa laidassa esittää konservatiivisen tason ja pistekatkoviiva realistisemman kynnysarvon.*

4 Lähdesignaalien tuottaminen

Tässä osiossa käsitellään vähentävän synteesin lähdesignaalien tuottamiseen käytettäviä oskillaattorialgoritmeja eli toisin sanoen menetelmiä toteuttaa digitaalisesti osiossa 2 esiteltyjä aaltomuotoja. Oskillaattorialgoritmien tutkimuksessa ja kehittämisessä laskostumisen välttäminen on ollut merkittävässä roolissa. Ideaalisessa tapauksessa laskostumista ei esiinny ollenkaan, mutta käytännön kannalta tietty määrä laskostumista voidaan kuitenkin sallia ilman, että se vaikuttaa syntyvään kuulokokemukseen.

Triviaalisesti tuotettua aaltomuotoa audiotekniikalle tyypillisillä näytteenottotaajuuksilla suoraan näytteistämällä saadaan pahasti laskostunut signaali, joka ei äänenlaatunsa puolesta yleensä kelpaa käytettäväksi musiikissa [4]. Siksi on täytynyt kehittää digitaalisia menetelmiä, joilla laskostuminen on vähäisempää tai sitä ei tapahdu. Olemassa olevat oskillaattorialgoritmit voidaan jakaa neljään eri ryhmään, jotka ovat ideaalisesti laskostuttomat, melkein laskostuttomat, laskostumisesta vaimentavat sekä ad-hoc-algoritmit. [2, 5]

4.1 Ideaalisesti laskostumattomat algoritmit

Ideaalisesti laskottumattomiksi algoritmeiksi voidaan luokitella menetelmät, jotka tuottavat vain halutun määrän harmonisia taajuuskomponentteja niin, että kaikki komponentit jäävät Nyquist-taajuuden alapuolelle. Koska Nyquist-taajuutta ei ylitetä, laskostumista ei pääse tapahtumaan. Näillä algoritmeillä on mahdollista saavuttaa erittäin hyvä äänenlaatu ja synnytettävien harmonisten komponenttien määrään on helppo vaikuttaa. Kääntöpuolena kategorian menetelmät voivat olla laskennallisesti erittäin raskaita ja sisältää laskennallisia ongelmia tai vaatia suuren määrän muistia. Laskostumattomien algoritmien käytännön toteutukset ovat näin ollen usein kompromisseja äänenlaadun ja laskennallisen tehokkuuden välillä. Yksinkertaisin laskostumaton menetelmä on hyödyntää additiivisen synteesin periaatetta, jota käsitellään seuraavassa luvussa. [2, 5]

4.1.1 Additiivinen synteesi

Haluttu määrä harmonisia komponentteja voidaan tuottaa suoraviivaisesti additiivisella synteesillä. Koska kaikki jaksolliset signaalit ovat esitettävissä siniaaltojen summana, halutut geometriset aaltomuodot voidaan esittää sarjana sini -tai kosiniaaltoja [7]. Additiivisessa synteesissä siis nimenmukaisesti lasketaan yhteen yksittäisiä ääneksiä eri taajuuksilla ja amplitudeilla, jotka yhdessä muodostavat haluttua signaalia vastaavan aaltomuodon ja spektrin. Käsiteltävien aaltomuotojen äärettömät Fourier-sarjat esitettiin kappaleessa 2. Laskemalla vain n ensimmäistä sarjan komponenttia voidaan tuottaa mielivaltainen määrä aaltomuodon harmonisia komponentteja, jolloin saamme kaistarajoitetun approksimaation aaltomuodon ideaalisesta tapauksesta. Kaistarajoitetut Fourier-sarjat saadaan korvaamalla ääretön summa

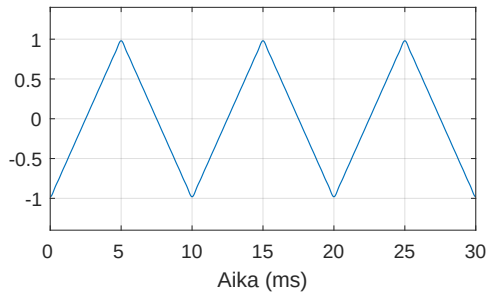
kaavoissa (1), (2) ja (3) äärellisellä indeksillä n, missä $n \in \mathbb{N}$.

$$x_{kolmio}(t) = -\frac{4}{\pi} \sum_{k=1}^{n} \frac{|\text{sinc}(\frac{k}{2})|}{k} \cos(2\pi k f t) \tag{4}$$

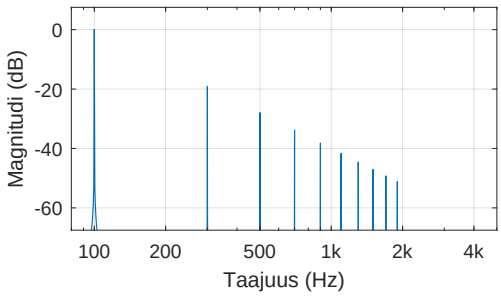
$$x_{suorakaide}(t) = \frac{4}{\pi} \sum_{k=1}^{n} \frac{\sin(2\pi(2k-1)ft)}{(2k-1)} \tag{5}$$

$$x_{saha}(t) = -\frac{2}{\pi} \sum_{k=1}^{n} \frac{\sin(2\pi k f t)}{k} \tag{6}$$

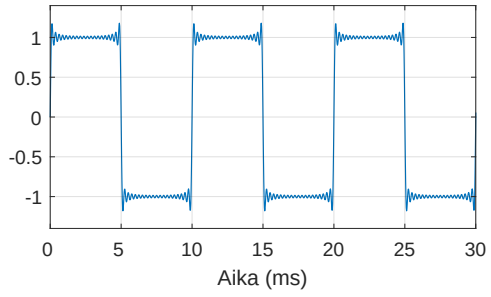
Kuvassa 5 on esitetty kaistarajoitetut aaltomuodot ja niitä vastaavat spektrit kaavoilla (4), (5) ja (6) tuotettuina, kun summaa lasketaan vain indeksiin $n = 20$ asti.



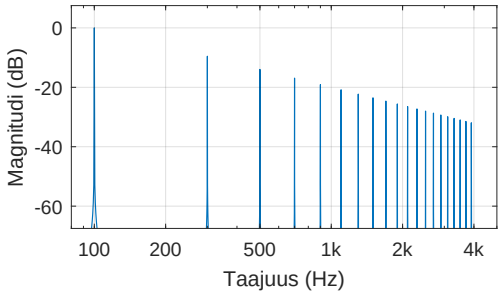
(a) Kolmioaalto



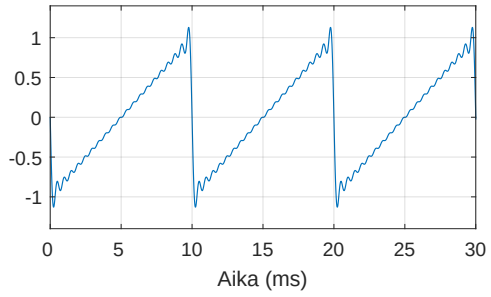
(b) Kolmioaallon spektri



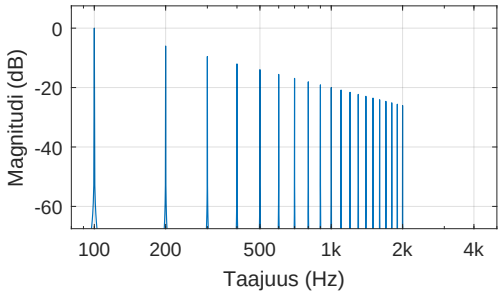
(c) suorakaideaalto



(d) suorakaideaallon spektri



(e) Saha-aalto



(f) Saha-aallon spektri

Kuva 5: *Geometriset aaltomuodot summakaavoilla tuotettuina, $n = 20$, $f = 100$ Hz.*

Kuvan 5 kohdissa (c) ja (e) nähdään niin sanottu Gibbsin ilmiö: epäjatkuvuuskohtiin syntyy äärellisellä määrällä taajuuskomponentteja noin yhdeksän prosentin huippuarvon ylitys, mikä johtuu pohjimmiltaan siitä, että yritetään esittää epäjatkuvaa signaalia äärellisellä määrällä jatkuvia signaaleita [7].

Additiivista synteesiä käyttämällä voidaan laskostumiselta välttyä kokonaan, mutta menetelmä on laskennallisesti raskas etenkin kun lasketaan suuri määrä taajuuskomponentteja. Tuotettavien komponenttien määrä on kääntäen verrannollinen perustaajuuteen, kun lasketaan kaikki Nyquist-taajuuden alle jäävät komponentit. Näin ollen laskennallinen kuorma ei pysy vakiona vaan kasvaa perustaajuuden pienentyessä. Tuotettavien komponenttien määrää voidaan rajata esimerkiksi lopettamalla sarjan laskeminen, kun määrätty vaimennustaso suhteessa perustaajuuteen on saavutettu tai laskemalla vaikka vain kymmenen ensimmäistä komponenttia, millä voidaan pienentää laskennallista taakkaa huomattavasti. Aaltomuodoille saadaan usein riittävän hyvä approksimaatio jo varsin pienellä määrällä taajuuskomponentteja. [2, 4, 7]

Additiivinen synteesi on mahdollista toteuttaa myös taajuusalueesta liikkeelle lähtien käyttämällä käänteistä Fourier-muunnosta. Tässä tapauksessa tuotetaan ensiksi haluttua signaalia vastaava spektri, jolle lasketaan niin sanottu nopea Fourier-muunnos (*fast Fourier transform, FFT*) käänteisenä. Näin saadaan tuotettua taajuuskomponentteja vastaava aikasignaali. [2, 8]

4.1.2 Aaltotaulukkosynteesi

Additiivisen menetelmän laskennallinen raskaus voidaan välttää käyttämällä aaltotaulukkosynteesiä (*wavetable synthesis*). Funktioiden jaksollisuuden ansiosta riittää, että lasketaan etukäteen valmiiksi yksittäisiä jaksoja aaltomuodosta eri perustaajuuksilla, ja tallennetaan ne taulukoihin. Lukemalla taulukkoa peräkkäin saadaan aikaiseksi jatkuva signaali, jota voidaan edelleen moduloida aikamuuttuvilla parametreilla mielenkiintoisemman äänen aikaansaamiseksi. Taajuutta voidaan muuttaa siirtymällä aaltotaulukoiden välillä tai vaihtoehtoisesti lukemalla taulukkoa eri nopeudella. Aaltotaulukkosynteesi tarjoaa myös yksinkertaisen oikotien todenmukaiseen analogisynteesiin: oikean analogisyntetisaattorin oskillaattorien tuottama signaali voidaan nauhoittaa ja tallentaa suoraan aaltotaulukkoon. [1, 2]

Aaltotaulukkosynteesi on lähdesignaalin tuottamisen kannalta sinänsä laskennallisesti tehokas menetelmä etenkin additiiviseen synteesin verrattuna, mutta sen sijaan muistivaatimukset kasvavat nopeasti suuriksi, kun aaltotaulukoita on paljon. Lisäksi aaltotaulukkosynteesin käytännön toteutus ei ole täysin ongelmaton, vaan menetelmä vaatii interpolaatiota näytteiden lukemiseen, kun haluttu taajuus ei ole kokonaislukukerroin aaltotaulukkoon tallennetusta perustaajuudesta. Interpolaatioalgoritmin laadusta riippuen interpolointi voi aiheuttaa kuultavia häiriöitä lopputulokseen. [1, 2, 8]

4.1.3 DSF

Kaistarajoitettu sarja siniääniä voidaan esittää vaihtoehtoisesti kahden sinifunktion suhteena, jolloin haluttu määrä komponentteja voidaan laskea samanaikaisesti yhdellä kaavalla ja laskutoimituksella. Menetelmästä käytetään nimitystä diskreetti summauskaava DSF (*discrete summation formula*). Laskutapa perustuu geometrisen sarjan summaan [8, 9]

$$\sum_{k=0}^{N-1} z^k = \frac{1 - z^N}{1 - z} \tag{7}$$

Tällä kaavalla laskettuna kaikkien harmonisten komponenttien amplitudi on kuitenkin sama, kun todellisuudessa spektrin tulisi olla laskeva kuten kuvassa 5 nähtiin. Täten DFS-menetelmällä tuotettua signaalia joudutaan alipäästösuodattamaan oikeanlaisen spektrin saavuttamiseksi. Myös jakolaskuoperaatio itsessään aiheuttaa ongelmia, kun jakajana on nolla tai hyvin lähellä nollaa oleva arvo. Tämä monimutkaistaa menetelmän käyttöä, kun nämä jakolaskuun liittyvät ongelmatilanteet joudutaan tarkastelemaan erikseen. [2, 4]

4.2 Melkein laskostumattomat algoritmit

Geometristen aaltomuotojen epäjatkuvuuskohtia voidaan pehmentää, eli pyöristää teräviä kulmia aaltomuodoissa, jolloin laskostuminen on vähäisempää. Tähän kategoriaan kuuluvien algoritmien voidaan siis käsittää tuottavan alipäästösuodatettuja versioita ideaalisista aaltomuodoista. Kun lisäksi muistetaan osiossa 3 esitellyt laskostumisen havaitsemiseen liittyvät psykoakustiset tekijät, algoritmeissa voidaan sallia jokin määrä laskostumista etenkin korkeilla taajuuksilla ilman, että se haittaa kuultavasti lopputulosta. Vanhin näistä menetelmistä on seuraavaksi käsiteltävä BLIT. [2, 4]

4.2.1 BLIT

Tim Stilson ja Julius Smith esittelivät vuonna 1996 kaistarajoitettuun impulssijonoon (*bandlimited impulse train*) perustuvan oskillaattorialgoritmin [8]. Tiedetään, että derivoimalla kolmioaaltoa ajan suhteen saadaan suorakaideaaltoa, ja edelleen suorakaideaalto ja saha-aalto kertaalleen derivoimalla jäljelle jää jono aaltomuotojen epäjatkuvuuskohdissa sijaitsevia yksikköimpulsseja $\delta(t)$, missä

$$\delta(t) = \begin{cases} 1 & , \text{kun } t = 0 \\ 0 & , \text{muulloin.} \end{cases} \tag{8}$$

Yksikköimpulssijono voidaan esittää summana viivästettyjä yksikköimpulsseja

$$x(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT), \tag{9}$$

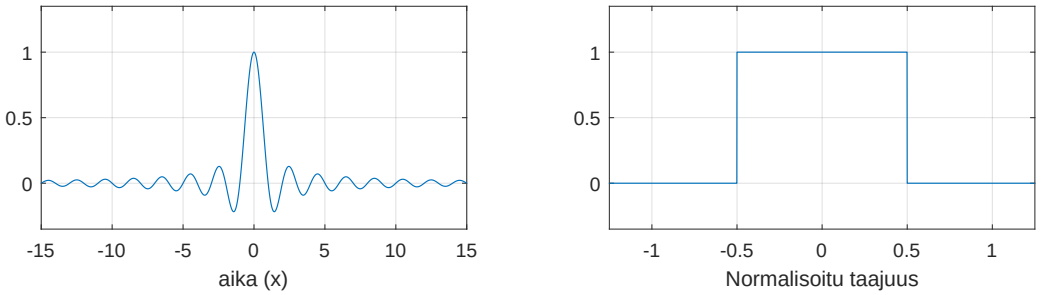
missä T on jaksonaika.

Tästä impulssijonosta saadaan kaistarajoitettu alipäästösuodattamalla se, mikä voidaan toteuttaa konvoloimalla impulssijono jonkin alipäästösuodattimen impulssivasteen $h_s(t)$ kanssa:

$$x_s(t) = (x * h_s)(t) = \sum_{k=-\infty}^{\infty} h_s(t - kT) \tag{10}$$

Kaavasta 10 nähdään, että impulssijonon tapauksessa konvoluutio vastaa suoraan jokaisen impulssin korvaamista suodattimen impulssivasteella. Integroimalla suodatettua impulssijonoa saadaan tuotettua likimäärin kaistarajoitettuja aaltomuotoja. Suorakaide- ja saha-aallot saadaan kertaalleen integroimalla sopivaa impulssijonoa ja kolmioaalto kahdesti integroimalla. [1, 2, 8]

BLIT -synteesissä käytetään alipäästösuodattimena tyypillisesti sinc-funktiota, joka vastaa ideaalista alipäästösuodatinta taajuustasossa, sillä suorakaidepulssin käänteisenä Fourier-muunnoksena saadaan aikatason sinc-funktio. Nämä esitetään kuvassa 6. Koska sinc-funktio on äärettömän pitkä, täytyy se ikkunoida käytännön toteutuksia varten. Ikkunoinnissa funktiosta muodostetaan äärellisen pituinen versio ottamalla siitä halutun levyinen kaistale, joka painotetaan ikkunafunktion muodolla. Tyypillisesti tämä ikkunoitu sinc-funktio esilasketaan valmiiksi taulukkoon. Tästä menetelmän erikoistapauksesta käytetään nimitystä BLIT-SWS (*sum of windowed sincs*). [1, 8]



(a) Sinc-funktio

(b) Suorakulmainen pulssi

Kuva 6: *Ideaalinen alipäästösuodatin (a) aikatasossa ja (b) taajuustasossa. Taajuusakselilla nähdään normalisoitu taajuus $f_N = f/f_s$, jolloin Nyquist-taajuus saa arvon $0,5$ ja näytteenottotaajuus arvon 1.*

BLIT-menetelmässä integrointi voi tuottaa ongelmia numeerisista epätarkkuuksista johtuen, kun impulssijonon arvot poikkeavat tarkoista arvoista esimerkiksi pyöristysvirheiden takia. Ongelma voidaan korjata käyttämällä toisen asteen vuotavaa integraattoria [8, 10]. BLIT-algoritmia voidaan parantaa edelleen monella tavalla, esimerkiksi optimoimalla suodatinfunktiota ja sen esilaskettua taulukkoa [2, 11]. Taulukko voidaan myös jakaa monivaihesuodatinrakenteeksi [1]. Tarve esilasketulle taulukolle voidaan välttää kokonaan hyödyntämällä murtoviiveisiä suodattimia (*fractional delay filter*) ikkunoidun sinc-funktion sijaan, joita käyttämällä voidaan myös vähentää laskostumista merkittävästi (BLIT-FDF) [12].

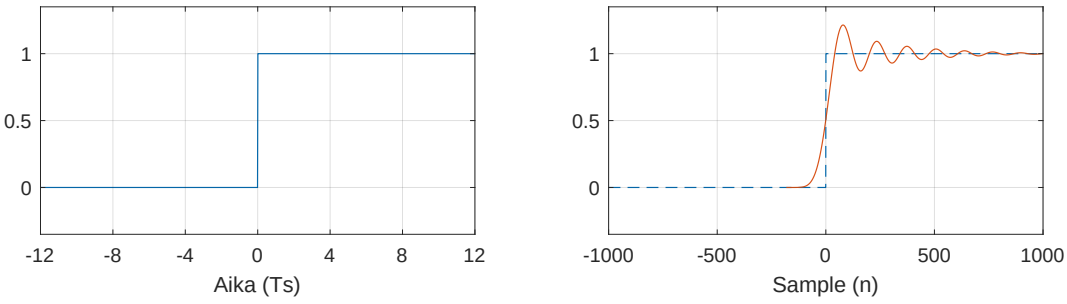
4.2.2 BLEP

Brandt esitteli vuonna 2001 BLIT-menetelmään pohjautuvan kehittyneemmän oskillaattorialgoritmin [10]. Geometriset aaltomuodot voidaan muodostaa myös yksikköaskelfunktion $u(x)$ avulla, missä

$$u(x) = \int_{-\infty}^{x} \delta(t)\,\mathrm{d}t = \begin{cases} 0 & , x < 0 \\ 0.5 & , x = 0 \\ 1 & , x > 0 \end{cases} \tag{11}$$

Näin ollen BLIT:n numeeriset ongelmat voidaan välttää integroimalla kaistarajoitettu impulssi etukäteen, jolloin saamme kaistarajoitetun askelfunktion (BLEP, *bandlimited step function*). Korvaamalla jokainen askelmainen epäjatkuvuuskohta aaltomuodossa tällä kaistarajoitetulla askelfunktiolla saadaan lopputuloksena likimäärin kaistarajoitettu signaali. [11]

Brandt ehdotti myös ikkunoidun sinc-funktion korvaamista sen amplitudivastetta vastaavalla minimivaiheisella FIR -suodattimella, joka integroimalla saadaan minimivaiheinen BLEP -funktio. Menetelmästä käytetään nimitystä MinBLEP (*minimum-phase bandlimited step*) [10]. Kuvassa 7 esitetään yksikköaskelfunktio ja minimivaiheinen BLEP.



(a) Askelfunktio (b) MinBLEP

Kuva 7: *(a) Yksikköaskelfunktio $u(t)$ ja (b) minimivaiheinen kaistarajoitettu askelfunktio. Aika näyteajan $Ts = 1/Fs$ moninkertana.*

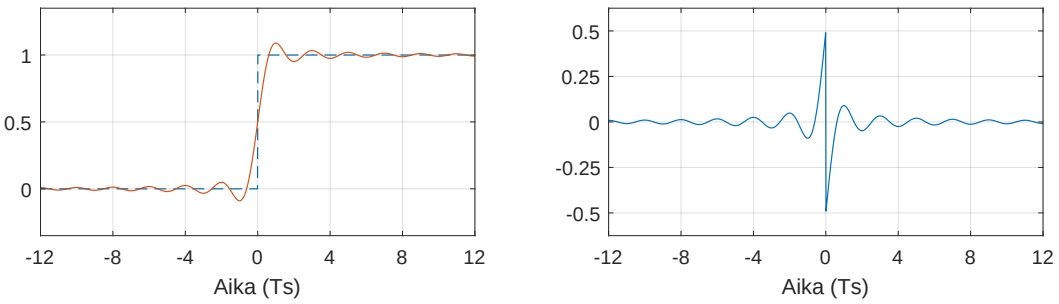
Välimäki et al. johtivat ideaalisen kaistarajoitetun askelfunktion, joka voidaan esittää sini-integraalin avulla [11]

$$h_{Is}(t) = \frac{1}{2} + \frac{1}{\pi}\mathrm{Si}(\pi\alpha f_s t), \tag{12}$$

missä $\alpha \in [0, 1]$ on skaalauskerroin rajataajuudelle ja

$$Si(x) = \int_0^x \frac{\sin(t)}{t}\,\mathrm{d}t = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!} \tag{13}$$

Käytännön toteutuksia varten tämä ideaalinen BLEP -funktio täytyy ikkunoida, sillä se on äärettömän pituinen sinc-funktion tapaan. Laskennallisesti tehokas tapa toteuttaa algoritmi on hyödyntää ideaalisen BLEP -funktion residuaalia eli jäännösvirhettä, joka saadaan vähentämällä ideaalisesta BLEP -funktiosta yksikköaskelfunktio. Ikkunoinnin jälkeen tämä jäännösvirhe voidaan lisätä suoraan triviaalisesti tuotetun aaltomuodon epäjatkuvuuskohtiin, jolloin saadaan likimäärin kaistarajoitettu versio aaltomuodosta. Laskevalla askeleella residuaali täytyy kuitenkin kääntää ylösalaisin. Ideaalinen kaistarajoitettu askelfunktio ja jäännösvirhe esitetään kuvassa 8. [1, 2, 11]



(a) Ideaalinen BLEP

(b) Jäännösvirhe

Kuva 8: *(a) Ideaalinen kaistarajoitettu askelfunktio $h_{Is}(t)$ ja yksikköaskel $u(t)$ (katkoviivalla), (b) jäännösvirhe $h_{Is}(t) - u(t)$. Aika näyteajan $Ts = 1/Fs$ moninkertana.*

BLEP-menetelmä voidaan myös toteuttaa muodostamalla kaistarajoitettu approksimaatio yksikköaskeleesta integroimalla polynomisia suodatinfunktioita, joista saadaan muodostettua BLEP-residuaali aikaisempaan tapaan. Menetelmästä käytetään nimitystä PolyBLEP (*polynomial bandlimited step function*), ja suodatinfunktioina käytetään polynomisia interpolaatiofunktioita. Yksinkertainen approksimaatio saadaan lineaarisella interpolaatiolla, joka vastaa kolmiomaista pulssia, ja parempiin tuloksiin päästään kasvattamalla polynomin astelukua. Menetelmä on yleistetty mielivaltaiselle polynomin asteluvulle, kun käytetään Lagrangen tai B-splini–interpolaatiota. Neljännen asteen Lagrange- ja B-splini–polynomeilla päästään laskostumisvapaaseen tulokseen yli viiden kilohertsin perustaajuuksille saakka, mikä riittää hyvin kattamaan kaikki perinteisen 88-koskettimisen pianon perustaajuudet, sillä pianon korkeimman sävelen C_8 perustaajuus on noin 4,2 kHz. [1, 2, 7, 11]

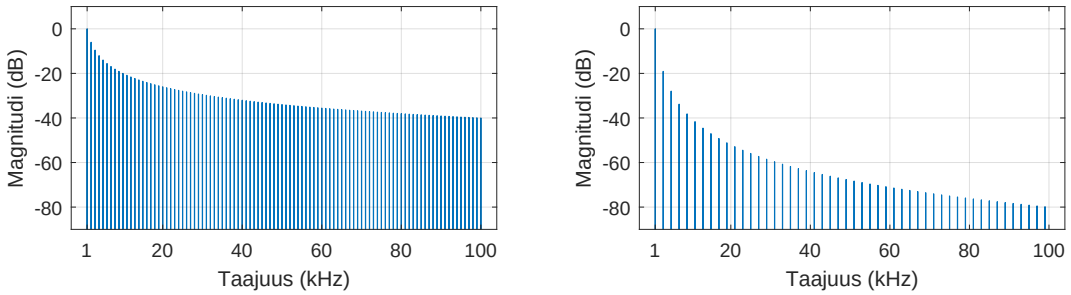
4.3 Laskostumista vaimentavat algoritmit

Laskostumista vaimentavat algoritmit poikkeavat kahdesta edellisestä kategoriasta siinä, että ne sallivat laskostumista koko audiokaistalla. Laskostumisen estämisen sijaan näissä algoritmeissä pyritään vaimentamaan laskostuneiden komponenttien voimakkuutta muokkaamalla spektrin muotoa ennen näytteistämistä, jolloin parhaassa tapauksessa laskostumista ei kuulla. Spektrin muokkaaminen vääristää tietysti aaltomuotoa, joten näytteistämisen jälkeen spektri täytyy palauttaa oikeanlaiseksi sopivalla ylipäästösuodattimella. Laskostumista vaimentavien menetelmien esiasteena voidaan pitää triviaalisesti tuotettujen aaltomuotojen ylinäytteistämistä eli

näytteenottotaajuuden moninkertaistamista, mitä käsitellään seuraavassa osiossa.

4.3.1 Ylinäytteistäminen

Triviaalisti tuotetut aaltomuodot laskostuvat pahasti etenkin saha -ja suorakaideaallon osalta, koska niiden taajuuskomponentit vaimenevat hitaasti, eli toisin sanoen aaltomuotojen spektri laskee hyvin loivasti. Siksi Nyquist-taajuuden ylittävillä komponenteilla on vielä suhteellisen suuri magnitudi, mikä johtaa tasoltaan suuriin laskostuneisiin komponentteihin. Kuvassa 9 nähdään saha- ja kolmioaallon taajuuskomponenttien vaimeneminen aina 100 kHz saakka. Saha- ja suorakaideaallolla spektri laskee kuusi desibelia oktaavilla (taajuuden kaksinkertaistuminen) eli 20 dB dekadia (taajuuden kymmenkertaistuminen) kohden. Kolmioaallon spektri vaimenee sen sijaan kaksi kertaa jyrkemmin, eli 12 dB oktaavilla ja 40 dB dekadilla.



(a) (b)

Kuva 9: *(a) Saha-aallon ja (b) kolmio-aallon taajuusvasteen vaimeneminen lineaarisella taajuusasteikoilla,* $n = 100$, $f = 1$ *kHz,* $F_s = 262$ *kHz.*

Näytteenottotaajuutta kasvattamalla Nyquist-taajuuden ylittävien komponenttien magnitudi pienenee, jolloin myös laskostuneiden komponenttien taso laskee. Kuvan 9 perusteella näytteenottotaajuuden kaksinkertaistaminen laskee laskostumisen tasoa vain 6 dB:n verran saha- ja suorakaideaallon tapauksessa. Tästä seuraa, että pelkkä ylinäytteistäminen itsessään on hyvin epätehokas menetelmä laskostumisen vaimentamiseen. Hyvän vaimennussuhteen saavuttaminen saha -ja suorakaideaallolla vaatisi erittäin suuren näytteenottotaajuuden, esimerkiksi kuvan 9 saha-aallon tapauksessa 60 dB vaimennus saavutetaan vasta 1 MHz taajuuskomponentilla, joka puolestaan vaatisi 2 MHz:n näytteenottotaajuuden, mikä ei ole toteutettavissa käytännön sovelluksissa.

Kolmioaallon tapauksessa puolestaan laskostumista esiintyy huomattavasti vähemmän jyrkemmän spektrin ansioista. Käytännössä jo 44,1 kHz:n näytteenottotaajuudella alle yhden kilohertsin perustaajuuksilla laskostuminen on melko vähäistä. Tämän kaksinkertaisella ylinäytteistyksellä, eli näytteenottotaajuudella 88,2 kHz laskostuneiden komponenttien taso jää hyvin pieneksi yli neljän kilohertsin perustaajuuksille saakka, mikä riittää kattamaan yleensä kaikki musikaalisesti kiinnostavat perustaajuudet. Kolmioaallon tapauksessa jo kohtuullisella ja helposti toteuttettavissa

olevalla ylinäytteistämisellä saadaan triviaalin tapauksen laskostumista vaimennettua huomattavasti. Ylinäytteistämisellä saatavia tuloksia esitellään yksityiskohtaisemmin osiossa 5.1.

Yksi ensimmäisistä varsinaisista spektrin muokkaamiseen perustuvista menetelmistä oli Lane et al. vuonna 1997 esittelemä saha-aaltoalgoritmi, jossa kokoaaltotasasuunnataan sini-aalto, jonka taajuus on puolet halutusta perustaajuudesta. Saadun signaalin spektri laskee saha-aaltoa jyrkemmin, ja näytteistämisen jälkeen saha-aallon aaltomuotoon päästään takaisin käyttämällä perustaajuudesta riippuvaa ylipäästösuodatinta sekä kiinteän taajuuden alipäästösuodatinta. Suorakaide -ja kolmioaalto voidaan tuottaa saadusta saha-aallosta. [2, 13]

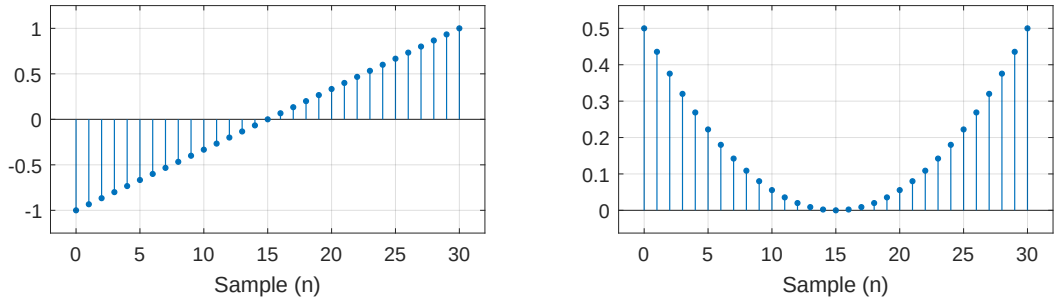
4.3.2 DPW

Välimäki esitteli uuden polynomisten aaltomuodon derivoimiseen perustuvan laskennallisesti tehokkaan menetelmän vuonna 2005 [13]. Alunperin menetelmällä voitiin tuottaa saha-aaltoa muistuttavaa aaltomuotoa derivoimalla paloittain määriteltyä parabolista aaltoa, ja sittemmin menetelmä on laajennettu myös muille aaltomuodoille ja suuremmille asteluvuille [3, 14]. Menetelmän nimi tulee sanoista differentioitu polynominen aaltomuoto (*differentiated polynomial waveform*).

Perusmenetelmässä aloitetaan integroimalla saha-aalto ajan suhteen, jolloin saadaan parabolisista osioista koostuva aaltomuoto. Saha-aallon yksi jakso voidaan esittää lineaarisena funktiona ajan suhteen, jolloin integraaliksi saadaan

$$p(t) = \int t\mathrm{d}t = \frac{t^2}{2}, \quad \text{kun } \frac{-T}{2} \leq t \leq \frac{T}{2} \tag{14}$$

missä T on yhden jakson pituus. Tämä esitetään kuvassa 10. [13]



(a) $t, -1 \leq t \leq 1$ (b) $p(t) = t^2/2$

Kuva 10: *(a) lineaarinen funktio ja (b) sen integraali ajan suhteen. Vaaka-akselin arvot näytteinä.*

Koska parabolisen aaltomuodon spektri laskee kaksi kertaa jyrkemmin kuin saha-aallon, saadaan laskostumista vaimennettua näytteistämällä tätä signaalia. Näytteistyksen jälkeen oikeanlainen spektri palautetaan ensiksi derivoimalla, jonka jälkeen signaali täytyy vielä skaalata kertoimella

$$c = \frac{f_s}{4f_0(1 - \frac{f_0}{f_s})}. \tag{15}$$

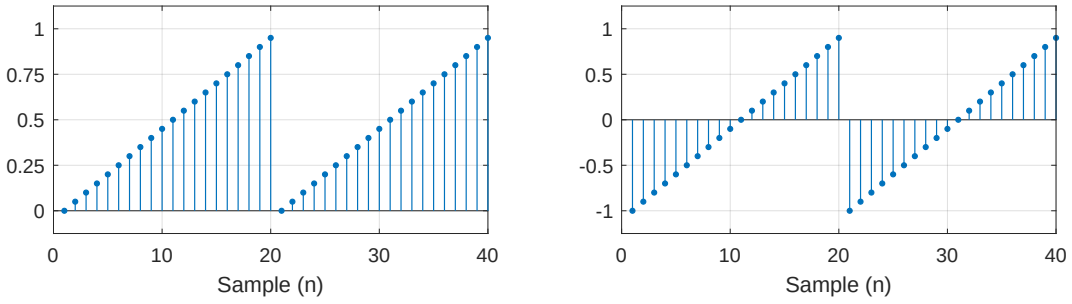
Yksinkertainen tapa tuottaa ideaalista saha-aaltoa digitaalisesti on hyödyntää jakojäännöstä. Diskreettiaikainen vaihesignaali voidaan muodostaa

$$p(n) = nf_0T \bmod 1, \tag{16}$$

josta edelleen saadaan välille -1 ja 1 skaalattu saha-aalto toteutettua niin sanottuna bipolaarisena jakojäännöslaskurina (*bipolar modulo counter*)

$$s(n) = 2p(n) - 1 = 2(nf_0T \bmod 1) - 1. \tag{17}$$

Vaihesignaali $p(n)$ ja jakojäännöslaskuri $s(n)$ esitetään kuvassa 11.



(a) $p(n)$ (b) $s(n)$

Kuva 11: *(a) Diskreettiaikainen vaihesignaali ja (b) jakojäännöslaskuri. Vaaka-akselin arvot näytteinä.*

Jakojäännöslaskurin signaali voidaan korottaa suoraan toiseen potenssiin parabolisen signaalin tuottamiseksi, ja lopuksi signaali derivoidaan ja skaalataan. Derivaattori voidaan toteuttaa eri tavoilla, joista yksinkertaisin on ensimmäisen asteen FIR-suodin siirtofunktiolla

$$h(z) = \frac{1 - z^{-1}}{2}, \tag{18}$$

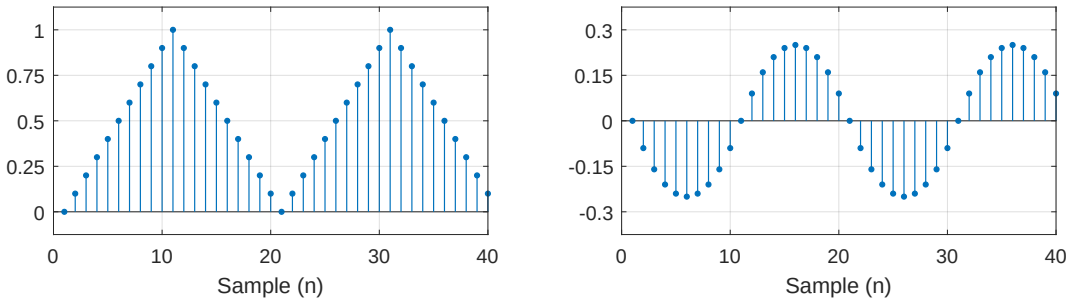
joka on niin sanottu ensimmäisen eron suodatin (*first-difference filter*). [3, 13]

Perusalgoritmia voidaan parantaa hyödyntämällä ylinäytteistystä, missä parabolinen aaltomuoto tuotetaan moninkertaisella näytteenottotaajuudella, jolloin laskostumista saadaan vaimennettua lisää. Lopuksi näytetaajuus lasketaan takaisin alkuperäiselle tasolle. Esimerkiksi kaksinkertaisen ylinäytteistyksen tapauksessa tämä voidaan toteuttaa ensiksi alipäästösuodattamalla uuden Nyquist-taajuuden ylittävät komponentit, jonka jälkeen joka toinen näyte jätetään välistä. [13]

Suorakulmaista aaltomuotoa, ja edelleen sen erikoistapauksena suorakaide-aaltoa, voidaan tuottaa vähentämällä kaksi sopivalla vaihe-erolla olevaa saha-aaltoa toisistaan. DPW:tä voidaan käyttää myös kolmioaallon tuottamiseen derivoimalla bipolaarista parabolista aaltomuotoa [3]. Tämä parabolinen aalto voidaan muodostaa jakojäännöslaskurin (kaava 17) avulla

$$s_b(n) = s(n)[1 - |s(n)|], \tag{19}$$

missä termi $[1 - |s(n)|]$ vastaa siirrettyä kokoaaltotasasuunnattua saha-aaltoa. Nämä esitetään kuvassa 12. Derivoimalla ja skaalaamalla tämä parabolinen aalto saadaan kolmiomaista aaltomuotoa vaimennetulla laskostumisella. [14]



(a) $1 - |s(n)|$ (b) $s_b(n)$

Kuva 12: *(a) Siirretty kokoaaltotasasuunnattu saha-aalto ja (b) bipolaarinen parabolinen aalto. Vaaka-akselin arvot näytteinä.*

Välimäki et. al laajensivat DPW-menetelmän suurempiastelukuisille polynomeille vuonna 2010 [14]. Laskostumista voidaan vaimentaa tehokkaammin käyttämällä suurempia astelukuja, sillä jokainen asteluvun kasvatus jyrkentää polynomisen aaltomuodon spektriä aina 6 dB. Vastaavasti jokaista asteluvun korotusta kohti tarvitaan yksi derivointi lisää sekä uusi skaalauskerroin. Sopiva mielivaltaisen asteluvun polynomi voidaan johtaa analyyttisesti integroimalla kaavaa (14), tai ratkaisemalla suoraan polynomiyhtälön kertoimet tunnettujen ehtojen avulla. Suurempiastelukuiset polynomiset aaltomuodot voidaan johtaa sekä saha- että kolmioaallolle. Lisäksi huomattiin, että suorakaide-aalto voidaan tuottaa suoraan kolmiomaisesta aaltomuodosta sopivasti käsittelemällä ja derivoimalla, joka pätee myös suuremmille asteluvuille. [14, 15]

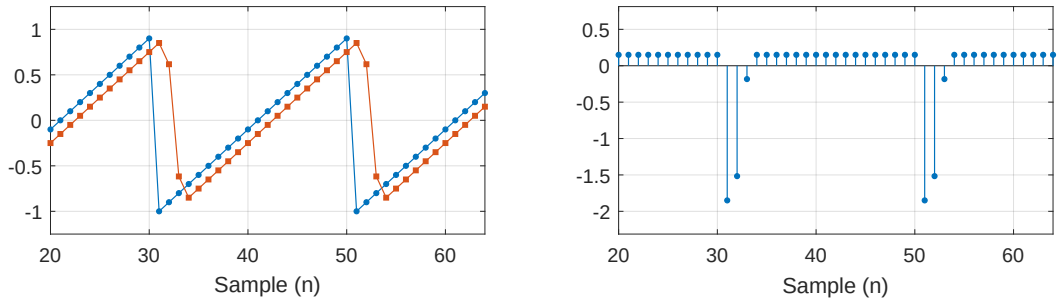
Suurempien astelukujen kanssa voidaan vielä lisäksi käyttää ylinäytteistämistä perusmenetelmän tapaan. Ylinäytteistämällä laskostumista saadaan vähennettyä etenkin matalilla taajuuksilla perustaajuuden alapuolella, ja ylinäytteistyksellä saatava hyöty kasvaa asteluvun mukana. Käytännössä ylinäytteistystä ei kuitenkaan yleensä kannata käyttää, koska sen tuoma lisäys laskennalliseen kuormaan on usein suurempi kuin asteluvun kasvattamisen aiheuttama kuorma ja asteluvun kasvatus vaimentaa laskostumista tehokkaammin kuin näytetaajuuden kaksinkertaistaminen. [14]

Laskennallisen analyysin perusteella saha-aallon tapauksessa kaksiasteisella perus-DPW:llä laskostuminen saattaa tulla kuuluviin yli 600 Hz:n perustaajuuksilla, kun näytteenottotaajuus on 44,1 kHz. Jo neljännen asteen DPW on puolestaan havainnon kannalta laskostumisvapaa noin 4,6 kHz:n perustaajuudelle saakka, mikä riittää yleensä hyvin käytännön sovelluksissa. Neljännen asteen DPW-oskillaattorilla saatavia tuloksia esitellään tarkemmin osiossa 5.2. Laskennallisesti tehokas tapa toteuttaa laskostumisvapaa DPW-oskillaattori on käyttää kaksi-asteista algoritmia pienien perustaajuuksien tuottamiseen ja siirtyä neljännen asteen versioon korkeampia perustaajuuksia varten. Suositeltu siirtymätaajuus astelukujen välillä on noin 500 Hz. [14]

4.3.3 PTR

Kleimola ja Välimäki huomasivat DPW-algoritmin muokkaavan ideaalista saha-aaltoa vain rajatulla alueella epäjatkuvuuskohdan ympärillä muiden näytteiden ollessa vain tasoltaan hieman siirtyneitä [16]. Lisäksi epäjatkuvuuskohdan ympärille tehdyt muutokset vastaavat muodoltaan suoraan polynomia. Näiden havaintojen perusteella DPW-menetelmästä voidaan tuottaa uusi vähemmän laskutoimituksia vaativa ja monipuolisempi oskillaattorialgoritmi, jota kutsutaan polynomisien siirtymäalueiden algoritmiksi (*polynomial transition regions*). [15, 16]

DPW-algoritmissä aaltomuodon kahden peräkkäisen jakson välinen siirtymäalue alkaa jakojäännösoperaation kohdalta (askelmainen epäjatkuvuuskohta), ja siirtymäalueen leveys $W = N - 1$ näytettä, missä N on DPW:n asteluku. Kuvassa 13(a) nähdään triviaali saha-aalto jakojäännöslaskurilla toteutettuna ja neljännen asteen DPW-algoritmillä tuotettu saha-aalto sekä kuvassa 13(b) näiden kahden erotus.



(a) (b)

Kuva 13: *(a) Triviaali saha-aalto (ympyrä) ja neljännen asteen DPW saha-aalto (neliö), sekä (b) näiden kahden erotus. Kuvassa $f_0 = 2400$ Hz ja $f_s = 48$ kHz.*

Kuvissa havaitaan teorian mukainen siirtymäalue $W = 3$, jolla DPW aaltomuodon näytearvot poikkeavat triviaalista aaltomuodosta. Siirtymäalueen ulkopuolella olevilla näytteillä havaitaan pelkästään vakioarvoinen vaihesiirtymä (*offset*), jonka arvo on $c_{dc} = WT_0$, missä W on siirtymäaluen leveys näytteinä ja $T_0 = f_0/f_s$. [16]

Näin ollen DWP-algoritmin tuottama saha-aaltosignaali voidaan jakaa kahteen osioon, jotka ovat polynomiset siirtymäalueet ja vakiolla siirretty triviaali saha-aalto

$$y(n) = \begin{cases} s(n) - c_{dc} + D(n) & , \text{kun } p(n) < WT_0 \\ s(n) - c_{dc} & , \text{kun } p(n) \geq WT_0 \end{cases} \tag{20}$$

missä $D(n)$ korjauspolynomi [17]. Yleinen PTR-algoritmi voidaan määritellä

$$y(n) = \begin{cases} x(n) - c_{dc} + D(n) & , \text{kun } p(n) < WT_0 \\ x(n) - c_{dc} & , \text{kun } p(n) \geq WT_0 \end{cases} \tag{21}$$

missä $x(n)$ on tulosignaali. PTR-menetelmässä riittää siis, että triviaalisesti tuotettua sisääntulosignaalia muokataan korjauspolynomilla vain siirtymäalueen sisällä, ja muulloin voidaan käyttää suoraan triviaalia signaalia vaihesiirrettynä. Koska siirtymäalueen pituus on yleensä jaksonpituutta paljon pienempi, laskutoimituksien määrää saadaan vähennettyä huomattavasti DPW:hen verrattuna, missä algoritmia lasketaan jokaiselle näytteelle. Vaadittavien aritmeettisien laskutoimituksien määrää tarkastelemalla PTR:lla saavutetaan yli 40 prosentin parannus DPW:hen nähden. DPW:n toteutuksilla saavutettava hyvä laskostumisen vaimentuminen pysyy PTR-menetelmässä ennallaan. PTR-algoritmia voidaan käyttää saha-aallon lisäksi muille aaltomuodoille, myös ei-derivoituville, toisin kuin DPW:ssä. Menetelmää voidaan soveltaa myös muihin käyttötarkoituksiin ja synteesimenetelmiin vähentävän synteesin lisäksi. [15, 16]

Ambrits ja Bank osoittivat, että PTR-menetelmän laskennallista tehokkuutta voidaan parantaa vielä lisää eliminoimalla tarve vaihesiirrolle. Menetelmästä käytetään nimitystä EPTR eli tehokas PTR (*efficient PTR*). Esimerkiksi toisen asteen DPW-algoritmin tuottama saha-aalto on puoli näytettä jäljessä triviaalisesta aallosta, jolloin yksinkertaisin tapa välttää vaihesiirto PTR-algoritmissä on käyttää tulosignaalina puoli näytettä haluttua signaalia edellä olevaa triviaalia saha-aaltoa, ja näin yhteenlaskua termin c_{dc} kanssa ei tarvita. EPTR-menetelmällä tuotetut aaltomuodot ovat identtisiä DPW:llä ja PTR:llä tuotettujen kanssa, mutta laskutoimituksien määrää voidaan vähentää vielä 30 prosenttia PTR:ään verrattuna. [18]

4.4 Ad-hoc-algoritmit

Ad-hoc (lat. *tätä varten*)–algoritmeihin lukeutuu useita erilaisia menetelmiä, joita yhdistää tunnettujen ja usein yksinkertaisten digitaalisen signaalinkäsittelyn menetelmien hyödyntäminen geometrisiä aaltomuotoja muistuttavien signaalien tuottamiseksi. Kategorian menetelmät saattavat laskostua algoritmista riippuen. Useat näistä perustuvat epälineaarisiin menetelmiin, kuten aaltomuotoiluun (*waveshaping*) tai vaihesäröön (*phase distortion*). [2, 15, 19]

5 Simulointi

Tässä osiossa tarkastellaan triviaalisesti tuotettuja aaltomuotoja ylinäytteistämällä ja neljännen asteen DPW-algoritmilla saatavia tuloksia. Simulointi toteutettiin MATLAB-ohjelmistolla ja tuotettu ohjelmakoodi löytyy liittestä A.

5.1 Ylinäytteistäminen

Tutkitaan triviaalisesti tuotetun kolmio- ja saha-aallon laskostumista eri näytteenottotaajuuksilla. Perustaajuuksina käytetään pianon säveliä C5, C6, C7 ja C8, joita vastaavat perustaajuudet on esitetty taulukossa 1.

Taulukko 1: *Perustaajuudet (Hz).*

523	1047	2093	4186

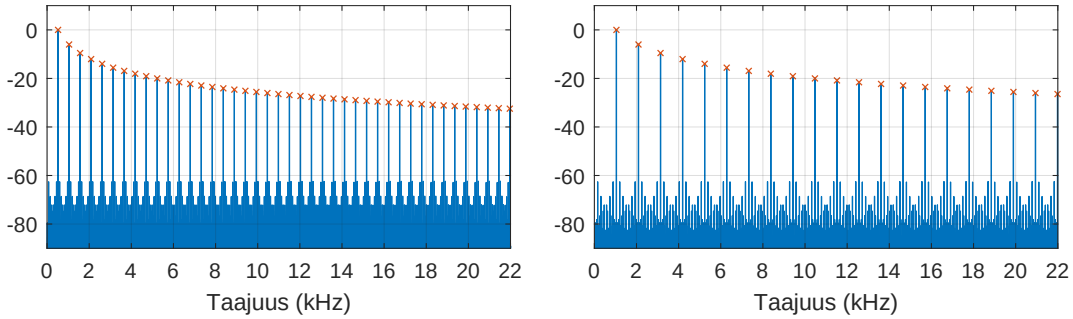
Saha-aallon tapauksessa tutkittiin, kuinka moninkertainen ylinäytteistys vaadittaisiin 44,1 kHz perusnäytteenottotaajuudelle, jotta kaikkien laskostuneiden komponenttien taso olisi vähintään 60 dB pienempi verrattuna perustaajuuteen. Kuvassa 14 esitetään jokaiselle taulukon 1 perustaajuudelle tarvittava näytteenottotaajuus, kun näytteenottotaajuus aina kaksinkertaistettiin. Tulokset esitetään taulukossa 2. Tulokset osoittavat hyvin, miksi saha-aallon ja samalla jyrkkyydellä laskevan suorakaide-aallon tapauksessa triviaalin menetelmän ylinäytteistäminen ei ole vaihtoehto käytännön toteutuksissa tarvittavan näytteenottotaajuuden ollessa useita megahertsejä.

Taulukko 2: *Vaadittava ylinäytteistys saha-aallolla.*

f_0 (Hz)	523	1047	2093	4186
f_s (kHz)	705,6	1 411,2	2 822,4	5 644,8
Ylinäytteistyskerroin	16	32	64	128

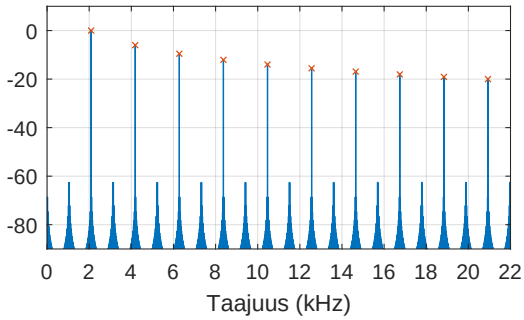
Kolmioaallon tapauksessa puolestaan tarkastellaan audiojärjestelmissä yleisintä 44,1 kHz:n näytteenottotaajuutta ja sen kaksinkertaista ylinäytteistettyä tapausta 88,2 kHz. Kiinnostuksen kohteena on kuinka paljon triviaali kolmioaalto laskostuu näillä näytteenottotaajuuksilla, jonka perusteella voidaan arvioida milloin triviaalin menetelmän äänenlaatu riittäisi mahdollisesti musiikkisovelluksiin. Jokaiselle perustaajuudelle laskettiin spektri kummallakin näytteenottotaajuudella. Saadut spektrit esitetään kuvassa 15.

Laskostuneiden komponenttien tason nähdään olevan yli 60 dB perustaajuutta matalampi yhden kilohertsin perustaajuuksille saakka molemmilla näytteenottotaajuuksilla. Ylinäytteistyksen tapauksessa tämä tilanne säilyy aina perustaajuudelle 2093 Hz saakka, ja 4186 Hz:n tapauksessakin vain muutama komponentti rikkoo kyseisen vaimennussuhteen. Tämän perusteella triviaalin kolmioaallon voidaan ar-

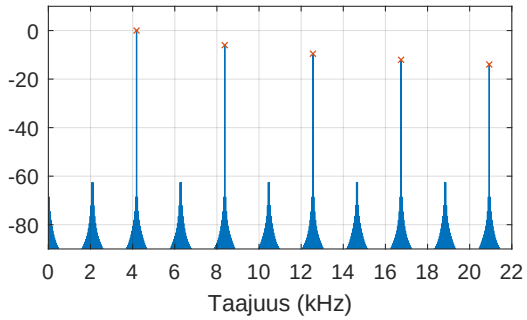


(a) $f_0 = 523$ Hz, $f_s = 705{,}6$ kHz

(b) $f_0 = 1047$ Hz, $f_s = 1\ 411{,}2$ kHz



(c) $f_0 = 2093$ Hz, $f_s = 2\ 822{,}4$ kHz



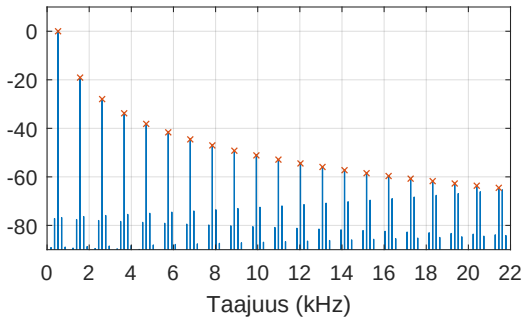
(d) $f_0 = 4186$ Hz, $f_s = 5\ 644{,}8$ kHz

Kuva 14: *Saha-aallon ylinäytteistäminen. Puhtaat spektrin komponentit ovat merkitty rasteilla.*

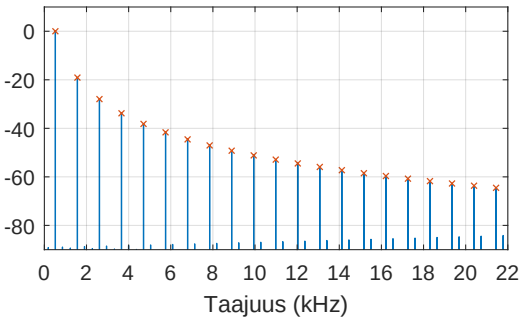
voida tarjoan kohtuullisen hyvän äänenlaadun etenkin pienillä perustaajuuksilla ja kaksinkertaisella ylinäytteistyksellä.

5.2 DPW

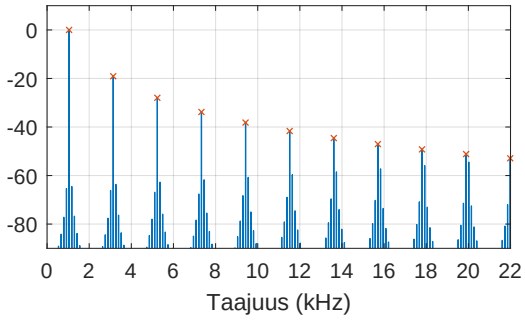
Osiossa tarkastellaan neljännen asteen DPW-algoritmilla saatavia tuloksia saha-aallolle taulukon 1 perustaajuuksilla. DPW4-algoritmin simulointi toteutettiin MATLAB-ohjelmistolla algoritmin kehittäjien jakaman Python-ohjelmointikielisen lähdekoodin pohjalta [17]. Kuvassa 16 nähdään jokaiselle tarkasteltavalle perustaajudelle oskillaattorialgoritmilla saatu aaltomuoto näytteinä sekä näitä vastaavat taajuusvasteet. Spektreissä rastit kuvaavat ideaalisen aaltomuodon taajuuskomponentteja. Kuvan perusteella voidaan todeta neljännen asteen DPW-algoritmin vaimentavan laskostumista tehokkaasti etenkin keski- ja alataajuuksilla. Lopputuloksena saatavan spektrin muoto ei aivan täysin vastaa ideaalista aaltomuotoa, mutta kuulokokemuksen kannalta ero on lähes mitätön.



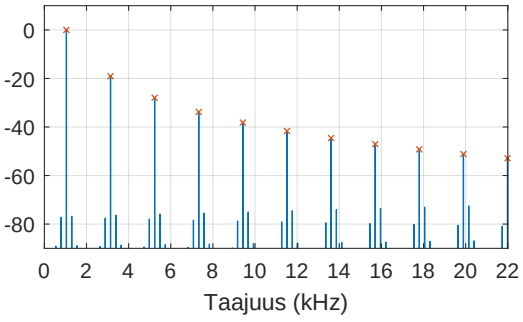
(a) $f_0 = 523$ Hz, $f_s = 44{,}1$ kHz



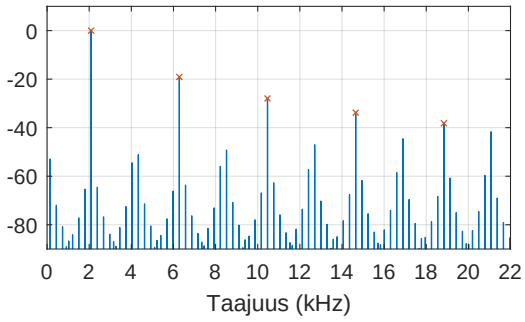
(b) $f_0 = 523$ Hz, $f_s = 88{,}2$ kHz



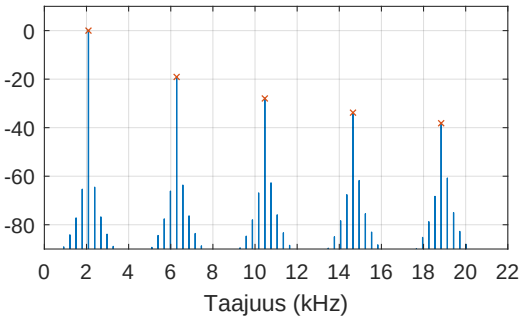
(c) $f_0 = 1047$ Hz, $f_s = 44{,}1$ kHz



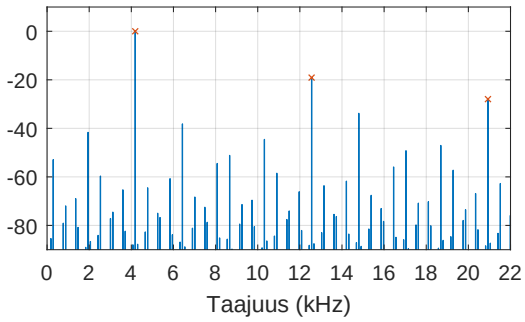
(d) $f_0 = 1047$ Hz, $f_s = 88{,}2$ kHz



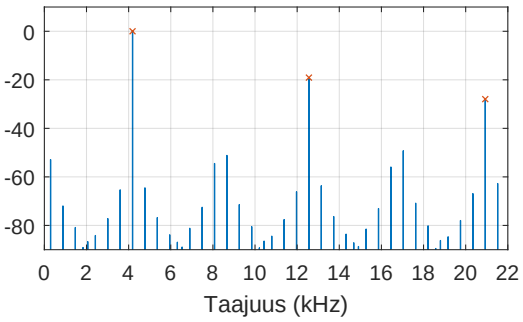
(e) $f_0 = 2093$ Hz, $f_s = 44{,}1$ kHz



(f) $f_0 = 2093$ Hz, $f_s = 88{,}2$ kHz

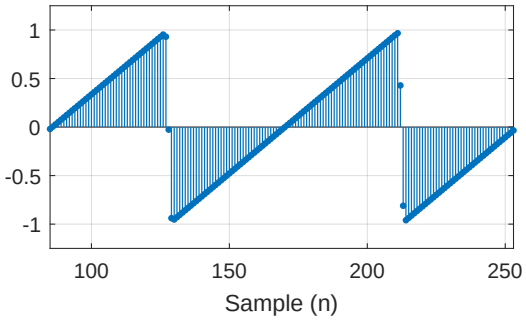


(g) $f_0 = 4186$ Hz, $f_s = 44{,}1$ kHz

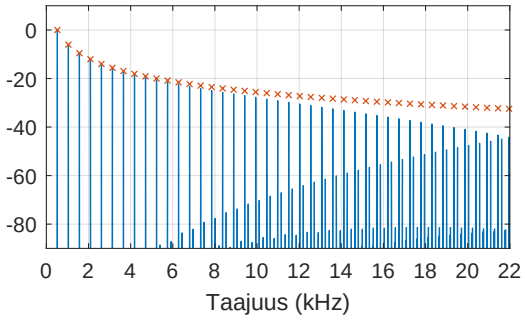


(h) $f_0 = 4186$ Hz, $f_s = 88{,}2$ kHz

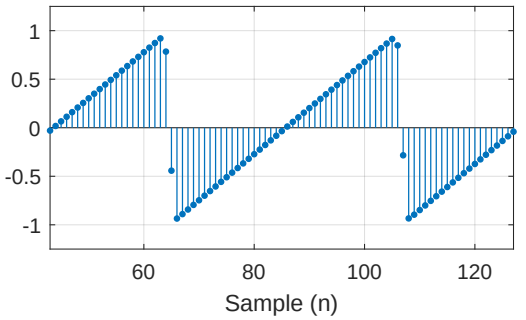
Kuva 15: *Triviaalin kolmioaallon spektrit lineaarisella taajuusasteikolla.*



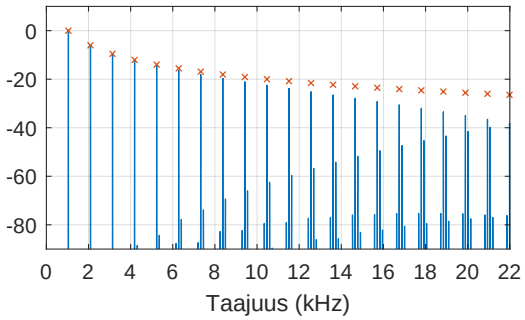
(a) $f_0 = 523$ Hz, näytteet.



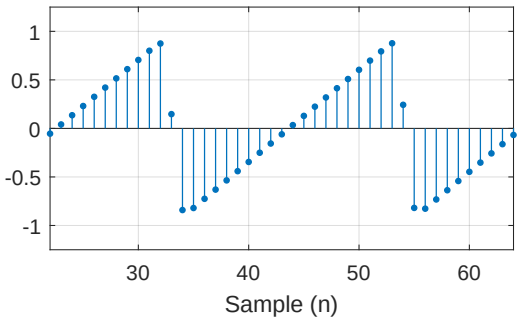
(b) $f_0 = 523$ Hz, spektri.



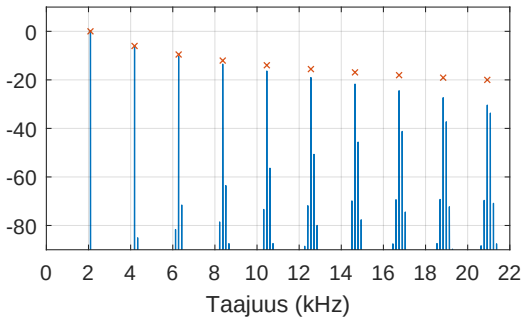
(c) $f_0 = 1047$ Hz, näytteet.



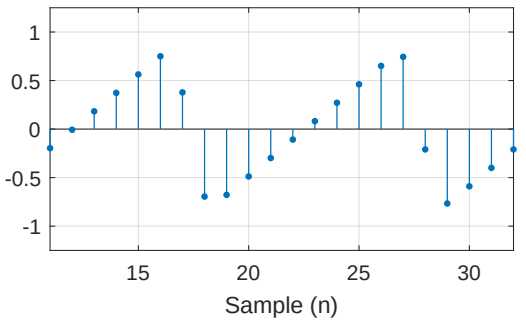
(d) $f_0 = 1047$ Hz, spektri.



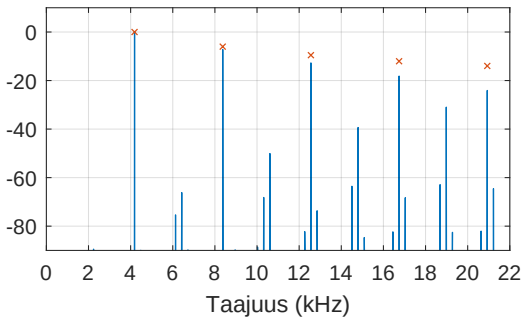
(e) $f_0 = 2093$ Hz, näytteet.



(f) $f_0 = 2093$ Hz, spektri.



(g) $f_0 = 4186$ Hz, näytteet.



(h) $f_0 = 4186$ Hz, spektri.

Kuva 16: *DPW-algoritmin simulaatio. Vasemmalla puolella kahden jakson verran näytteitä ja oikealla vastaava spektri.*

6 Yhteenveto

Tässä työssä esiteltiin useita erilaisia menetelmiä vähentävässä synteesissä tarvittavien lähdesignaalien digitaaliseen tuottamiseen. Optimaalisen oskillaattorialgoritmin ominaisuudet ovat:

1. algoritmin tuottamassa äänessä ei kuulla laskostumista musiikissa käytetyillä perustaajuuksilla (20–8000 Hz),
2. menetelmä on laskennallisesti tehokas ja vaatii vähän muistia, sekä
3. algoritmissa ei tarvita jakolaskua aikamuuttuvalla parametrilla.

Yksikään tässä työssä esitellyistä menetelmistä ei täytä kaikkia kolmea vaatimusta. Yhtä täydellistä oskillaattorialgoritmia ei siis ole ainakaan vielä olemassa, vaan käytettävä menetelmä joudutaan valitsemaan käyttötarkoituksen ja haluttujen ominaisuuksien perusteella. Esitellyt menetelmät on koottu taulukkoon 3 käsittelyjärjestyksessä. Käytännön tilanteissa kuitenkin monien menetelmien vaatimukset ja äänenlaatu riippuvat voimakkaasti toteutustavasta sekä esimerkiksi käytettävästä asteluvusta, joten taulukossa esitetyt arvot ovat suuntaa antavia. [1, 2, 5, 20]

Taulukko 3: *Oskillaattorialgoritmien ominaisuuksien vertailu.*

Algoritmi	Laskennallinen kuorma	Muistivaatimukset	Äänenlaatu
Triviaali	erittäin pieni	-	heikko
Additiivinen	erittäin suuri	erittäin pieni	erinomainen
Aaltotaulukko	keskiverto - suuri	suuri	erittäin hyvä
BLIT-SWS	keskiverto - suuri	pieni	hyvä
BLIT-FDF	pieni	erittäin pieni	erittäin hyvä
BLEP	keskiverto	pieni	hyvä
PolyBLEP	pieni	erittäin pieni	erittäin hyvä
Lane	keskiverto	erittäin pieni	keskiverto
DPW	pieni	erittäin pieni	erittäin hyvä
PTR	pieni	erittäin pieni	erittäin hyvä
EPTR	erittäin pieni	erittäin pieni	erittäin hyvä

Varsinaisen digitaalisen vähentävän synteesin toteuttaminen vaatii tässä työssä käsitellyn lähdesignaalien tuottamisen lisäksi analogisten syntetisaattoreiden suodattimien mallintamisen. Työssä esitellyissä lähdesignaalien generointimenetelmissä on keskitytty tuottamaan mahdollisimman lähelle ideaalista tapausta vastaavia geometrisiä aaltomuotoja. Kuitenkin todellisuudessa monien analogisyntetisaattorien tuottamat aaltomuodot poikkeavat ideaalisista tapauksista, jolloin realistiseen lopputulokseen vaaditaan näiden poikkeamien mallintamista [20]. Tähän voidaan soveltaa suodatinmallinnuksessa käytettyjä signaali- ja piirimallinnusmenetelmiä, jolloin lähimmäs todellisuutta päästään mallintamalla yksittäisen analogisyntetisaattorin sähköinen toiminta myös lähdesignaalien osalta. [5]

Viitteet

[1] Välimäki, Vesa ja Huovilainen, Antti: *Antialiasing Oscillators in Subtractive Synthesis*. IEEE Signal Processing Magazine, 24(2):116–125, 2007.

[2] Pekonen, Jussi: *Filter-Based Oscillator Algorithms for Virtual Analog Synthesis*. Väitöskirja, Aalto-yliopisto, Signaalinkäsittelyn ja akustiikan laitos, Espoo, 2014.

[3] Välimäki, Vesa ja Huovilainen, Antti: *Oscillator and Filter Algorithms for Virtual Analog Synthesis*. Computer Music Journal, 30(2):19–31, 2006.

[4] Välimäki, Vesa ja Huovilainen, Antti: *Virtuaalista nostalgiaa - digitaalinen vähentävä äänisynteesi*. Musiikki, 35(1-2):78–98, 2005. Suomen musiikkitieteellinen seura.

[5] Pekonen, Jussi ja Välimäki, Vesa: *Virtuaalianalogiasynteesin lyhyt historia*. Teoksessa *Akustiikkapäivät 2011*, sivut 45–50, Tampere, 2011. Akustinen Seura ry.

[6] Lehtonen, Heidi-Maria, Pekonen, Jussi ja Välimäki, Vesa: *Laskostumisen havaitseminen saha-aallossa*. Teoksessa *Akustiikkapäivät 2013*, Turku, 2013. Akustinen Seura ry.

[7] Rossing, Thomas D., Moore, F. Richard ja Wheeler, Paul A.: *The Science of Sound*. Addison-Wesley, 3. painos, 2001.

[8] Stilson, Tim ja Smith, Julius: *Alias-free digital synthesis of classic analog waveforms*. Teoksessa *Proc. International Computer Music Conference*, sivut 332–335, Hongkong, 1996.

[9] Lowenfels, David: *Virtual analog synthesis with a time-varying comb filter*. Teoksessa *Audio Engineering Society Convention 115*, New York, NY, 2003. Audio Engineering Society.

[10] Brandt, Eli: *Hard Sync Without Aliasing*. Teoksessa *International Computer Music Conference*, sivut 365–368, Havana, Kuuba, 2001.

[11] Välimäki, Vesa, Pekonen, Jussi ja Nam, Juhan: *Perceptually informed synthesis of bandlimited classical waveforms using integrated polynomial interpolation*. Journal of the Acoustical Society of America, 131(1):974–986, 2012.

[12] Nam, Juhan, Välimäki, Vesa, Abel, Jonathan S. ja Smith, Julius O.: *Efficient antialiasing oscillator algorithms using low-order fractional delay filters*. IEEE Transactions on Audio, Speech, and Language Processing, 18(4):773–785, 2010.

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

[14] Välimäki, V., Nam, J., Smith, J.O. ja Abel, J.S.: *Alias-Suppressed Oscillators Based on Differentiated Polynomial Waveforms*. IEEE Transactions on Audio, Speech, and Language Processing, 18(4):786–798, 2010.

[15] Kleimola, Jari: *Nonlinear Abstract Sound Synthesis Algorithms*. Väitöskirja, Aalto-yliopisto, Signaalinkäsittelyn ja akustiikan laitos, Espoo, 2013.

[16] Kleimola, Jari ja Välimäki, Vesa: *Reducing Aliasing from Synthetic Audio Signals Using Polynomial Transition Regions*. IEEE Signal Processing Letters, 19(2):67–70, 2012.

[17] Kleimola, Jari ja Välimäki, Vesa: *Polynomial Transition Regions*, 9.11.2011. Viitattu 14.12.14, saatavilla: http://research.spa.aalto.fi/publications/papers/spl-ptr.

[18] Ambrits, Daniel ja Bank, Balazs: *Improved Polynomial Transition Regions Algorithm For Alias-Suppressed Signal Synthesis*. Teoksessa *Proceedings of the Sound and Music Computing Conference*, sivut 561–568, Tukholma, Ruotsi, 2013.

[19] Timoney, Joseph, Lazzarini, Victor, Carty, Brian ja Pekonen, Jussi: *Phase and Amplitude Distortion Methods for Digital Synthesis of Classic Analog Waveforms*. Teoksessa *Audio Engineering Society Convention 126*, München, Saksa, 2009.

[20] Pekonen, Jussi, Lazzarini, Victor, Timoney, Joseph, Kleimola, Jari ja Välimäki, Vesa: *Discrete-time modelling of the Moog sawtooth oscillator waveform*. EURASIP Journal on Advances in Signal Processing, 2011:3, 2011.

[21] Välimäki, Vesa: *Audiosignaalinkäsittelyn sanasto*, 12.2.2010. Viitattu 1.12.2014, saatavilla: http://users.spa.aalto.fi/vpv/ask-sanasto.htm.

A MATLAB ohjelmakoodi

Liitteenä MATLAB -ohjelmistolle kirjoitetut komentosarjat, joilla tuotettiin kaikki työn kuvaajat paitsi 4.

aaltomuodot.m

```
papersize = [16 9]./1.6; paperpos  = [0 0 papersize]; screen = get(0,'screensize');

%% Kolmioaalto
x = 0:0.001:2*(2*pi); y = abs(4*(mod((x./(2*pi)-1/4),1))-2)-1;
figure('Position', [0, screen(4)-500, 800, 500]);
plot(x, y,'LineWidth',0.6); grid on; axis([0 4*pi -1.3 1.3]); xlabel('Jakso');
set(gca,'XTick',[0 1*pi 2*pi 3*pi 4*pi],'XTickLabel',{0 0.5 1 1.5 2});
set(gca,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\kolmioaalto','-dpdf','-painters');

%% suorakaideaalto
x = 0:0.001:2*(2*pi);
y = sign(sin(x));
figure('Position', [0, screen(4)-500, 800, 500]);
plot(x, y,'LineWidth',0.6); grid on; axis([0 4*pi -1.3 1.3]); xlabel('Jakso');
set(gca,'XTick',[0 1*pi 2*pi 3*pi 4*pi],'XTickLabel',{0 0.5 1 1.5 2});
set(gca,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\suorakaideaalto','-dpdf','-painters');

%% Saha-aalto
x = 0:0.001:4;
y = 2*((x+1)./2-floor((x+1)./2))-1;
figure('Position', [0, screen(4)-500, 800, 500]);
plot(x, y,'LineWidth',0.6); grid on; axis([0 4 -1.3 1.3]); xlabel('Jakso');
set(gca,'XTick',[0 1 2 3 4],'XTickLabel',{0 0.5 1 1.5 2});
set(gca,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\saha_aalto','-dpdf','-painters');

%% Kaanteinen sahalaita-aalto
x = 0:0.001:4;
y = -2*((x+1)./2-floor((x+1)./2))+1;
figure('Position', [0, screen(4)-500, 800, 500]);
plot(x, y,'LineWidth',0.6); grid on; axis([0 4 -1.3 1.3]); xlabel('Jakso');
set(gca,'XTick',[0 1 2 3 4],'XTickLabel',{0 0.5 1 1.5 2});
set(gca,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\saha_aalto_kaan','-dpdf','-painters');
%%
close all;
```

moogalipaasto.m

```
papersize = [16 9]./1.2; paperpos = [0 0 papersize]; screen = get(0,'screensize');

%% Moogin alipaastosuodattimen digitaalimallinnuksen taajuusvaste
Fs = 44100; % naytteenottotaajuus
Vt = 0.026; % Vt = 26 mV huoneenlammossa
k = 3.25; % takaisinkytkennan vahvistus
fc = 1200; % rajataajuus (ei vastaa todellista arvoa)
N = 44100; % naytteiden maara
i = zeros(N, 1); % Impulssivektori, pituus 1s
i(1) = 1; % ensimmainen arvo = 1
% Rajataajuden parametri
A = pi*(fc/Fs)*(1-(pi*(fc/Fs)))/(1+(pi*(fc/Fs)));
yi = zeros(N, 1); % ulostulostaulukko
% alustetaan muuttujat
[yT, sPrev, yPrev, I1, I2, I3, I4, yS1, yS2, yS3, yS4] = deal(0);
% suodatus yksi nayte kerrallaan
for n = 1:N
    yT = i(n) + k * yPrev;
    yT = (-1)*tanh(1/(2*Vt)*yT);
    % S1
    sPrev = yS1;
    yS1 = (yT - sPrev)*(2*A*Fs);
    I1 = I1 + (yS1 + sPrev)/(2*Fs);
    yS1 = tanh(I1);
    % S2
    sPrev = yS2;
```

```
    yS2 = (yS1 - sPrev)*(2*A*Fs);
    I2 = I2 + (yS2 + sPrev)/(2*Fs);
    yS2 = tanh(I2);
    % S3
    sPrev = yS3;
    yS3 = (yS2 - sPrev)*(2*A*Fs);
    I3 = I3 + (yS3 + sPrev)/(2*Fs);
    yS3 = tanh(I3);
    % S4
    sPrev = tanh(yS4/(2*Vt));
    yS4 = (yS3 - sPrev)*(2*A*Fs);
    I4 = I4 + (yS4 + sPrev)/(2*Fs);
    yS4 = I4*(2*Vt);
    yi(n) = yS4;
    yPrev = yS4;
end
Xf = 2*abs(fft(yi));
Xf = 20*log10(Xf./(Xf(1))); % normalisoidaan 0 dB:seen
f0 = Fs/N; tf = 0:f0:(N-1)*f0; % taajuusvektori

figure('Position', [0, screen(4)-450, 800, 450]);
semilogx(tf, Xf,'LineWidth',0.8); grid on;
ylabel('Vahvistus (dB)'); xlabel('Taajuus (Hz)');
axis([20 16000 -55 25]);
set(gca,'XMinorGrid','off','Layer','Top',...
        'XTick',[30 60 125 250 500 1000 2000 4000 8000 16000])
set(gca,'XTickLabel',{30 60 125 250 500 '1k' '2k' '4k' '8k' '16k'})
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\moogalipaasto','-dpdf','-painters');
%%
close all;
```

fouriersarjat.m

```
papersize = [16 9]./1.6; paperpos  = [0 0 papersize]; screen = get(0,'screensize');

f  = 100;                % perustaajuus
Fs = 44100;              % naytteenottotaajuus
Ts = 1/Fs; t = 0:Ts:2;   % aikavektori
N  = max(size(t));       % naytteiden maara

%% Kolmio
kolmio = zeros(1,N); % alustetaan taulukko
for k = 1:20 % lasketaan sarjaa
    kolmio = kolmio + abs(sinc(k/2))/k*cos(2*pi*k*f.*t);
end
kolmiosarja = kolmio*(-4)/pi;

% aaltomuoto
figure('Position',[0, screen(4)/2-250, 800, 500]);
plot(t, kolmiosarja,'LineWidth',0.6); grid on; axis([0 3/f -1.4 1.4]);
xlabel('Aika (ms)'); set(gca,'XtickLabel',0:5:30,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\kolmiosarja','-dpdf','-painters');

% taajuuspektri
Xf = 2*abs(fft(kolmiosarja)); % yksipuoleinen spektri
Xf = 20*log10(Xf./(max(Xf))); % skaalataan perustaajuus 0 dB:seen
f0 = Fs/N; tf = 0:f0:(N-1)*f0; % taajuusvektori
figure('Position',[0, 0, 800, 500]);
semilogx(tf, Xf,'LineWidth',0.6); grid on;
ylabel('Magnitudi (dB)'); xlabel('Taajuus (Hz)'); axis([80 5000 -67.5 7.5]);
set(gca,'XMinorGrid','off','Layer','Top',...
        'XTick',     [100 200 500 1000 2000 4000],...
        'XTickLabel',{100 200 500 '1k' '2k' '4k'});
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\kolmiospektri','-dpdf','-painters');

%% Suorakaide
kantti = zeros(1,N);
for k = 1:20
    kantti = kantti + sin(2*pi*(2*k-1)*f.*t)/(2*k-1);
end
kanttisarja = kantti*4/pi;

% aaltomuoto
figure('Position',[0, screen(4)/2-250, 800, 500]);
plot(t, kanttisarja,'LineWidth',0.6); grid on; axis([0 3/f -1.4 1.4]);
xlabel('Aika (ms)'); set(gca,'XtickLabel',0:5:30,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\suorakaidesarja','-dpdf','-painters');

% taajuuspektri
Xf = 2*abs(fft(kanttisarja));
Xf = 20*log10(Xf./(max(Xf)));
```

```
f0 = Fs/N; tf = 0:f0:(N-1)*f0;
figure('Position',[0, 0, 800, 500]);
semilogx(tf, Xf,'LineWidth',0.6); grid on; % Spektri
ylabel('Magnitudi (dB)'); xlabel('Taajuus (Hz)'); axis([80 5000 -67.5 7.5]);
set(gca,'XMinorGrid','off','Layer','Top',...
        'XTick',      [100 200 500 1000 2000 4000],...
        'XTickLabel',{100 200 500 '1k' '2k' '4k'});
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\suorakaidespektri','-dpdf','-painters');

%% Saha
saha = zeros(1,N);
for k = 1:20
    saha = saha + sin(2*pi*k*f.*t)/k;
end
sahasarja = -2/pi*saha;

% aaltomuoto
figure('Position',[0, screen(4)/2-250, 800, 500]);
plot(t, sahasarja,'LineWidth',0.6); grid on; axis([0 3/f -1.4 1.4]);
xlabel('Aika (ms)'); set(gca,'XtickLabel',0:5:30,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\sahasarja','-dpdf','-painters');

% taajuuspektri
Xf = 2*abs(fft(sahasarja));
Xf = 20*log10(Xf./(max(Xf)));
f0 = Fs/N; tf = 0:f0:(N-1)*f0;
figure('Position',[0, 0, 800, 500]);
semilogx(tf, Xf,'LineWidth',0.6); grid on; % Spektri
ylabel('Magnitudi (dB)'); xlabel('Taajuus (Hz)'); axis([80 5000 -67.5 7.5]);
set(gca,'XMinorGrid','off','Layer','Top',...
        'XTick',      [100 200 500 1000 2000 4000],...
        'XTickLabel',{100 200 500 '1k' '2k' '4k'});
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\sahaspektri','-dpdf','-painters');
%%
close all;
```

laskostuminen.m

```
papersize = [16 9]./1.6; paperpos = [0 0 papersize]; screen = get(0,'screensize');

%% Triviaali saha-aalto
f = 1000; % taajuus (Hz)
Fs = 44100; % naytteenottotaajuus
Ts = 1/Fs; t = 0:Ts:2-Ts; N = max(size(t));
saha = sawtooth(2*pi*f*t); % sahafunktio
Xf = 2*abs(fft(saha)); % yksipuoleinen spektri
Xf = 20*log10(Xf./(max(Xf))); % skaalataan perustaajuus 0 dB:seen
f0 = Fs/N; tf = 0:f0:(N-1)*f0; % taajuusvektori

figure('Position',[0, screen(4)/2-250, 800, 500]);
plot(tf, Xf,'LineWidth',0.6); grid on; axis([500 5500 -56.25 6.25]);
ylabel('Magnitudi (dB)'); xlabel('Taajuus (kHz)');
set(gca,'XTick',[1000 2000 3000 4000 5000])
set(gca,'XTickLabel',{1 2 3 4 5},'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\saha5lask','-dpdf','-painters');

figure('Position',[0, 0, 800, 500]);
plot(tf, Xf,'LineWidth',0.6); grid on; axis([0 20000 -56.25 6.25]);
ylabel('Magnitudi (dB)'); xlabel('Taajuus (kHz)');
set(gca,'XTick',[0 5000 10000 15000 20000]);
set(gca,'XTickLabel',{0 5 10 15 20}); set(gca, 'Layer', 'Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\saha20lask','-dpdf','-painters');

%% Laskostumattomat komponentit
% Fourier sarja, komponentteja vain 20 kHz asti
summa = zeros(1,N);
for k = 1:20
    summa = summa + sin(2*pi*k*f.*t)/k;
end
sahasarja = -2/pi*summa;
Xf = 2*abs(fft(sahasarja)); % yksipuoleinen spektri
Xf = 20*log10(Xf./(max(Xf))); % skaalataan perustaajuus 0 dB:seen
f0 = Fs/N; tf = 0:f0:(N-1)*f0; % taajuusvektori

figure('Position',[0, screen(4)/2-250, 800, 500]);
plot(tf, Xf,'LineWidth',0.6); grid on; axis([500 5500 -56.25 6.25]);
ylabel('Magnitudi (dB)'); xlabel('Taajuus (kHz)');
set(gca,'XTick',[1000 2000 3000 4000 5000])
set(gca,'XTickLabel',{1 2 3 4 5},'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
```

```
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\saha5','-dpdf','-painters');

figure('Position',[0, 0, 800, 500]);
plot(tf, Xf,'LineWidth',0.6); grid on; axis([0 20000 -56.25 6.25]);
ylabel('Magnitudi (dB)'); xlabel('Taajuus (kHz)');
set(gca,'XTick',[0 5000 10000 15000 20000]);
set(gca,'XTickLabel',{0 5 10 15 20},'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\saha20','-dpdf','-painters');
%%
close all;
```

sincfunktio.m

```
papersize = [16 9]./1.6; paperpos = [0 0 papersize]; screen = get(0,'screensize');

%% Sinc-funktio
t = -30:0.001:30; h = sinc(t);
figure('Position', [0, screen(4)/2-250, 800, 500]);
plot(t,h,'LineWidth',0.6); grid on; axis([-15 15 -0.35 1.35]);
xlabel('aika (x)'); set(gca,'XTick',-15:5:15);
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\sinc','-dpdf','-painters');

%% Suorakaidepulssi
t = -2:0.001:2; y = rectpuls(t);
figure('Position', [0, 0, 800, 500]);
plot(t,y,'LineWidth',0.6); grid on; axis([-1.25 1.25 -0.35 1.35]);
xlabel('Normalisoitu taajuus'); set(gca,'XTick',[-1 -0.5 0 0.5 1]);
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\rect','-dpdf','-painters');
```

askelfunktio.m

```
papersize = [16 9]./1.6; paperpos = [0 0 papersize]; screen = get(0,'screensize');
%% Askelfunktio
Fs = 44100;
t  = -12/Fs:1/(100*Fs):12/Fs;
y  = heaviside(t);
figure('Position', [0, screen(4)/2-500, 800, 500]);
plot(t,y,'LineWidth',0.6); grid on;
axis([-12/Fs 12/Fs -0.35 1.35]); xlabel('Aika (Ts)');
set(gca,'XTick',[-12/Fs -8/Fs -4/Fs 0 4/Fs 8/Fs 12/Fs]);
set(gca,'XTickLabel',{-12 -8 -4 0 4 8 12}); set(gca,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\askelfunktio','-dpdf','-painters');

%% Ideaalinen BLEP
Fs    = 44100;
t     = -16/Fs:1/(100*Fs):16/Fs;
iblep = 1/2 + 1/pi*sinint(pi*1*Fs*t);
istep = heaviside(t);
figure('Position', [0, screen(4)/2-500, 800, 500]);
plot(t, istep,'--','LineWidth',0.6); grid on; hold on;
plot(t, iblep,'LineWidth',0.6); hold off;
axis([-12/Fs 12/Fs -0.35 1.35]); xlabel('Aika (Ts)');
set(gca,'XTick',[-12/Fs -8/Fs -4/Fs 0 4/Fs 8/Fs 12/Fs]);
set(gca,'XTickLabel',{-12 -8 -4 0 4 8 12}); set(gca,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\blep','-dpdf','-painters');

%% BLEP residuaali
figure('Position', [0, screen(4)/2-500, 800, 500]);
plot(t, iblep-istep,'LineWidth',0.6); grid on;
axis([-12/Fs 12/Fs -0.625 0.625]); xlabel('Aika (Ts)');
set(gca,'YTick',-0.5:0.25:0.5);
set(gca,'XTick',[-12/Fs -8/Fs -4/Fs 0 4/Fs 8/Fs 12/Fs]);
set(gca,'XTickLabel',{-12 -8 -4 0 4 8 12}); set(gca,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\blepres','-dpdf','-painters');

%% minBLEP
load('minblep.mat','minblep'); % minBLEP funktio taulukoituna
% http://www.musicdsp.org/archive.php?classid=1#112
x  = linspace(-2000,2000,2*2048);
x2 = linspace(-182,2048-182, 2048); % sovitetaan 0-kohdat
figure('Position', [0, screen(4)/2-500, 800, 500]);
plot(x,heaviside(x),'--','LineWidth',0.6); grid on; hold on;
```

```
plot(x2, minblep,'LineWidth',0.6); hold off;
axis([-1000 1000 -0.35 1.35]); xlabel('Sample (n)'); set(gca,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\minblep','-dpdf','-painters');
%%
close all;
```

vaimeneminen.m

```
papersize = [16 9]./1.6; paperpos = [0 0 papersize]; screen = get(0,'screensize');

f  = 1000; % perustaajuus
Fs = 2^18; % naytteenottotaajuus (fft nopea kahden potensseille)
Ts = 1/Fs; t = 0:Ts:1-Ts; % aikavektori
N = max(size(t));

%% Saha-aalto
saha = zeros(1,N);
for k = 1:100
    saha = saha + sin(2*pi*k*f.*t)/k;
end
sahasarja = -2/pi*saha;
Xf = 2*abs(fft(sahasarja));
Xf = 20*log10(Xf./(max(Xf)));
f0 = Fs/N; tf = 0:f0:(N-1)*f0;

figure('Position',[0, screen(4)-500, 800, 500]);
plot(tf, Xf,'LineWidth',0.6); grid on; axis([-3000 104000 -90 10]);
ylabel('Magnitudi (dB)'); xlabel('Taajuus (kHz)');
set(gca,'XTick',[1000 20000 40000 60000 80000 100000])
set(gca,'XTickLabel',{'1' '20' '40' '60' '80' '100'},'Layer','Top')
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\saha100','-dpdf','-painters');

%% Kolmio-aalto
kolmio = zeros(1,N);
for k = 1:100
    kolmio = kolmio + abs(sinc(k/2))/k*cos(2*pi*k*f.*t);
end
kolmiosarja = kolmio*(-4)/pi;
Xf = 2*abs(fft(kolmiosarja));
Xf = 20*log10(Xf./(max(Xf)));
f0 = Fs/N; tf = 0:f0:(N-1)*f0;

figure('Position',[0, 0, 800, 500]);
plot(tf, Xf,'LineWidth',0.6); grid on; axis([-3000 104000 -90 10]);
ylabel('Magnitudi (dB)'); xlabel('Taajuus (kHz)');
set(gca,'XTick',[1000 20000 40000 60000 80000 100000])
set(gca,'XTickLabel',{'1' '20' '40' '60' '80' '100'},'Layer','Top')
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\kolmio100','-dpdf','-painters');
%%
close all;
```

phi.m

```
function s = phi(T0,p0,Fs)
% vaihesignaali p(n)
% http://research.spa.aalto.fi/publications/papers/spl-ptr/
    s = zeros(1,Fs); p = p0;
    for n = 1:Fs
        s(n) = p;
        p = p + T0;
        if p > 1
            p = p - 1;
        end
    end
end
```

dpw.m

```
papersize = [16 9]./1.6; paperpos = [0 0 papersize]; screen = get(0,'screensize');
dot = 2.5;

%% DPW
% saha-aalto ja parabolinen aaltomuoto
```

```
t = linspace(-1,1,31);  % suora -1...1
p = 1/2*t.^2;           % integraali
n = 0:1:max(size(t))-1;
figure('Position', [0, screen(4)/2-250, 800, 500]);
stem(n,t,'filled','lineWidth',0.6,'MarkerSize',dot); grid on;
axis([-2 32 -1.25 1.25]); grid on; xlabel('Sample (n)');
set(gca,'Xtick',0:5:30,'YTick',-1:0.5:1,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\sahastem','-dpdf','-painters');

figure('Position', [0, 0, 800, 500]);
stem(n,p,'filled','lineWidth',0.6,'MarkerSize',dot); grid on;
axis([-2 32 -0.0625 0.5625]); xlabel('Sample (n)');
set(gca,'Xtick',0:5:30,'YTick',0:0.1:0.5,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\parastem','-dpdf','-painters');

%% Kaksisuuntainen jakojaannoslaskuri
Fs = 48000;
f0 = 2400;  % perustaajuus
T0 = f0/Fs; % normalisoitu taajuus
P0 = Fs/f0;
mc = phi(T0,0,Fs);       % vaihesignaali
bmc = 2*phi(T0,0,Fs)-1; % kaksisuuntainen
figure('Position', [0, screen(4)/2-250, 800, 500]);
stem(mc,'filled','lineWidth',0.6,'MarkerSize',dot); grid on;
axis([0 ceil(2*P0) -0.125 1.125]); xlabel('Sample (n)');
set(gca,'YTick',0:0.25:1);
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\phi','-dpdf','-painters');

figure('Position', [0, 0, 800, 500]);
stem(bmc,'filled','lineWidth',0.6,'MarkerSize',dot); grid on;
axis([0 ceil(2*P0) -1.25 1.25]); xlabel('Sample (n)');
set(gca,'YTick',-1:0.5:1,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\bmc','-dpdf','-painters');

%% Kokoaaltotasasuunnattu saha-aalto
frs = 1-abs(2*phi(T0,0,Fs)-1);
figure('Position', [0, screen(4)/2-250, 800, 500]);
stem(frs,'filled','lineWidth',0.6,'MarkerSize',dot); grid on;
axis([0 ceil(2*P0) -0.125 1.125]); xlabel('Sample (n)');
set(gca,'YTick',0:0.25:1,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\frs','-dpdf','-painters');

%% Kaksisuuntainen parabolinen aaltomuoto
bp = (2*phi(T0,0,Fs)-1).*(1-abs(2*phi(T0,0,Fs)-1));
figure('Position', [0, 0, 800, 500]);
stem(bp,'filled','lineWidth',0.6,'MarkerSize',dot); grid on;
axis([0 ceil(2*P0) -0.375 0.375]); xlabel('Sample (n)');
set(gca,'YTick',-0.3:0.15:0.3,'Layer','Top');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\bp','-dpdf','-painters');
%%
close all;
```

ptr.m

```
papersize = [16 9]./1.6; paperpos = [0 0 papersize]; screen = get(0,'screensize');
dot = 2.5;

%% DPW ja triviaali ero
f0 = 2400;                    % perustaajuus
Fs = 48000;                   % naytteenottotaajuus
T0 = f0/Fs;                   % normalisoitu taajuus
P0 = Fs/f0; N = Fs;           % naytemaara
saha = 2*phi(T0,0.5,Fs) - 1; % triviaali saha
dpw  = DPW4(T0,P0,Fs);        % dpw saha

figure('Position',[0, screen(4)/2-250, 800, 500]);
plot(saha,'-o','LineWidth',0.6,'MarkerEdgeColor',[0 0.4470 0.7410],...
                               'MarkerFaceColor',[0 0.4470 0.7410],...
                               'MarkerSize',dot); grid on; hold on;
plot(dpw,'-s','LineWidth',0.6,'MarkerEdgeColor',[0.8500 0.3250 0.0980],...
                               'MarkerFaceColor',[0.8500 0.3250 0.0980],...
                               'MarkerSize',dot);
axis([ceil(P0) ceil(3.2*P0) -1.25 1.25]); xlabel('Sample (n)');
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\ptrhuomio','-dpdf','-painters');
```

```
% erotus
figure('Position', [0, 0, 800, 500]);
stem(saha-dpw,'filled','lineWidth',0.6,'MarkerSize',dot); grid on;
axis([ceil(P0) ceil(3.2*P0) -2.3125 0.8125]); xlabel('Sample (n)');
set(gca,'YTick',-2:0.5:0.5);
set(gcf,'PaperUnits','centimeters',...
        'PaperSize',papersize,'PaperPosition',paperpos);
print(gcf,'.\figures\ptrero','-dpdf','-painters');
```

ylinaytteistaminen.m

```
papersize = [16 9]./1.6; paperpos = [0 0 papersize]; screen = get(0,'screensize');
dot = 4;

%% Ylinaytteistaminen
ptaajuudet = [523 1047 2093 4186]; % perustaajuudet C5 C6 C7 C8
%% Kolmio-aalto
ntaajuudet = [44100 88200]; % naytteenottotaajuudet
for a = 1:4
    f = ptaajuudet(a);
    for b = 1:2
        Fs = ntaajuudet(b); Ts = 1/Fs; t  = 0:Ts:2-Ts;
        N  = max(size(t));  f0 = Fs/N; tf = 0:f0:(N-1)*f0;
        % triviaali kolmioaalto
        kolmio = 1-2*abs(sawtooth(2*pi*f*t));
        Xf = 2*abs(fft(kolmio)); Xf = 20*log10(Xf./(max(Xf)));

        % puhdaat komponentit
        kolmiop = zeros(1,N);
        for k = 1:floor(Fs/2/f)
            kolmiop = kolmiop + abs(sinc(k/2))/k*cos(2*pi*k*f.*t);
        end
        kolmiosarja = kolmiop*(-4)/pi;
        Xfp = 2*abs(fft(kolmiosarja)); Xfp = 20*log10(Xfp./(max(Xfp)));

        figure('Position',[0, screen(4)-500, 800, 500]);
        plot(tf, Xf,'LineWidth',0.6); grid on; hold on;
        plot(tf, Xfp,'x','MarkerSize',dot); hold off;
        axis([0 22000 -90 10]);
        xlabel('Taajuus (kHz)'); ylabel('Magnitudi (dB)');
        set(gca,'XTick', 0:2000:22000,'Layer','Top');
        set(gca,'XTickLabel',{0 2 4 6 8 10 12 14 16 18 20 22})
        nimi = strcat('.\figures\kolmio_',int2str(f),'hz_fs_',int2str(Fs));
        set(gcf,'PaperUnits','centimeters',...
            'PaperSize',papersize,'PaperPosition',paperpos);
        print(gcf,nimi,'-dpdf','-painters'); close;
    end
end

%% Saha-aalto
ntaajuudet = zeros(1,6); ntaajuudet(1) = 2*88200;
aste = zeros(1,6); aste(1) = ntaajuudet(1)/44100;
% naytteenottotaajuudet
for k = 2:6
    ntaajuudet(k) = 2*ntaajuudet(k-1);
    aste(k) = ntaajuudet(k)/44100;
end
for a = 1:4
    f = ptaajuudet(a);
    for b = 1:6
        Fs = ntaajuudet(b); Ts = 1/Fs; t = 0:Ts:2-Ts; N = max(size(t));
        f0 = Fs/N; tf = 0:f0:(N-1)*f0;
        % triviaali saha-aalto
        saha = sawtooth(2*pi*f*t);
        Xf = 2*abs(fft(saha)); Xf = 20*log10(Xf./(max(Xf)));

        % puhdaat komponentit
        sahap = zeros(1,N);
        for k = 1:floor(44100/f)
            sahap = sahap + sin(2*pi*k*f.*t)/k;
        end
        sahasarja = -2/pi*sahap;
        Xfp = 2*abs(fft(sahasarja)); Xfp = 20*log10(Xfp./(max(Xfp)));

        figure('Position',[0, 0, 800, 500]);
        plot(tf(1,1:52000), Xf(1,1:52000),'LineWidth',0.6); grid on; hold on;
        plot(tf(1,1:52000), Xfp(1,1:52000),'x','MarkerSize',dot); hold off;
        axis([0 22000 -90 10]);
        ylabel('Magnitudi (dB)'); xlabel('Taajuus (kHz)');
        set(gca,'XTick',0:2000:22000,'Layer','Top')
        set(gca,'XTickLabel',{0 2 4 6 8 10 12 14 16 18 20 22})
        set(gcf,'PaperUnits','centimeters',...
            'PaperSize',papersize,'PaperPosition',paperpos);
        print(gcf,strcat('.\figures\saha_',int2str(f),'hz_fs_',int2str(b)),...
                        '-dpdf','-painters'); close;
    end
end
```

DPW4.m

```
function y = DPW4(T0,P0,Fs)
% Neljannen asteen DPW algoritmi
% http://research.spa.aalto.fi/publications/papers/spl-ptr/
    [z0, z1, z2] = deal(0); % alustetaan muuttujat
    c = (P0*P0*P0)/192;     % skaalauskerroin
    p = phi(T0,0.5,Fs);     % vaihesignaali
    y = zeros(1,Fs);        % ulostulotaulukko
    for n = 1:Fs            % prosessointi nayte kerrallaan
        s  = 2*p(n) - 1;    % triviaali saha
        s2 = s*s;           % toisen asteen paraboli
        s4 = s2*(s2 - 2);   % neljannen asteen paraboli
        y0 = s4 - z0;       % derivointi 1
        y1 = y0 - z1;       % derivointi 2
        y2 = y1 - z2;       % derivointi 3
        y(n) = y2 * c;      % skaalaus
        z0 = s4;            % paivitetaan edelliset arvot
        z1 = y0;            %          derivointia varten
        z2 = y1;
    end
    y(1) = 0;   % asetetaan kolme ensimmaista naytetta nollaksi
    y(2) = 0;   %                 niihin syntyvan virheen takia
    y(3) = 0;
end
```

simulointi.m

```
papersize = [16 9]./1.6; paperpos  = [0 0 papersize]; screen = get(0,'screensize');
dot = 2.5;

%% Simulointi
ptaajuudet = [523 1047 2093 4186]; % perustaajuudet C5 C6 C7 C8
Fs = 44100; % naytteenottotaajuus
for i = 1:4
    % puhdas Fourier-sarja
    f0 = ptaajuudet(i); T0 = f0/Fs; P0 = Fs/f0; Ts = 1/Fs;
    t = 0:Ts:1-Ts;           % aikavektori 1s
    n = 0:1:max(size(t))-1; % naytevektori
    N = max(size(t));        % naytteiden maara
    sahasarja = zeros(1,N);
    for k = 1:floor(22050/f0)
        sahasarja = sahasarja + sin(2*pi*k*f0.*t)/k;
    end
    sahasarja = -2/pi*sahasarja;
    Xf = 2*abs(fft(sahasarja));          % yksipuoleinen spektri
    Xfsaha = 20*log10(Xf./(max(Xf)));    % skaalataan perustaajuus 0 dB:seen
    f = Fs/N; tfsaha = 0:f:(N-1)*f;      % taajuusvektori

    % puhtaat samplet
    %figure('Position',[0, screen(4)-500, 800, 500]);
    %stem(n,sahasarja,'filled','lineWidth',0.7,'MarkerSize',dot); grid on;
    %axis([ceil(P0) ceil(3*P0) -1.1 1.1]); xlabel('Sample (n)');

    % DPW
    dpw = DPW4(T0,P0,Fs); % neljannen asteen DPW
    figure('Position',[0, screen(4)/2-250, 800, 500]);
    stem(n,dpw,'filled','lineWidth',0.6,'MarkerSize',dot); grid on;
    axis([ceil(P0) ceil(3*P0) -1.25 1.25]); xlabel('Sample (n)');
    set(gcf,'PaperUnits','centimeters',...
            'PaperSize',papersize,'PaperPosition',paperpos);
    print(gcf,strcat('.\figures\dpw_',int2str(f0),'Hz_stem'),'-dpdf','-painters');

    Xf = 2*abs(fft(dpw));              % yksipuoleinen spektri
    Xf = 20*log10(Xf./(max(Xf)));    % skaalataan perustaajuus 0 dB:seen
    f = Fs/N; tf = 0:f:(N-1)*f;      % taajuusvektori
    figure('Position',[0, 0, 800, 500]);
    plot(tf, Xf,'LineWidth',0.6); grid on; axis([0 22000 -90 10]); hold on;
    ylabel('Magnitudi (dB)'); xlabel('Taajuus (kHz)');
    set(gca,'XTick', 0:2000:22000,'Layer','Top');
    set(gca,'XTickLabel',{0 2 4 6 8 10 12 14 16 18 20 22})
    plot(tfsaha, Xfsaha,'x','MarkerSize',4); hold off; % puhtaat komponentit
    set(gcf,'PaperUnits','centimeters',...
            'PaperSize',papersize,'PaperPosition',paperpos);
    print(gcf,strcat('.\figures\dpw_',int2str(f0),'Hz'),'-dpdf','-painters');
    close all;
end
```