Digital lydbehandling

Øyvind Hammer
Norsk nettverk for Teknologi, Akustikk og Musikk

Oslo, 5. april 1994

Innhold

Kapittel 1

Innledning

Dette heftet omhandler teknikker for å syntetisere, analysere og omforme lyd i digital form ved hjelp av datamaskiner. Det forutsettes at leseren har et visst grunnlag i akustikk og digital lyd, f.eks. gjennom heftene "Matematikk for musikere", "Akustikk for musikere" og "Sampling og rekonstruksjon".

Dette dokumentet er skrevet ved hjelp av typesettingsprogrammet LaTeX. Grafene er laget med matematikkprogrammet MATLAB. Tex-fil og Postscript-fil er tilgjengelige via anonym FTP på notam.uio.no (gjør `cd pub/doc/kurs`).

Kapittel 2

Lydbehandling og syntese i tidsdomenet

2.1 Amplityde- og ringmodulasjon

I analoge synthesizere benyttes spenningsstyrte forsterkere (VCA) til å endre amplityden til en lyd. En varierende kontrollspenning styrer da amplityden til lyden som passerer gjennom VCA'en, og vi sier at lyden *amplitydemoduleres* av kontrollsignalet (modulatoren).

I den digitale verdenen gjør vi dette ved å multiplisere hver av sample-verdiene i lyden med tilsvarende samples i kontrollsignalet. Hvis vi multipliserer med en konstant, vil hele lyden endre amplityde uniformt. Multiplikasjon med 0 gir total dempning, multiplikasjon med 0.5 gir 6 dB dempning, 1.0 endrer ikke amplityden, 2.0 gir 6 dB forsterking.

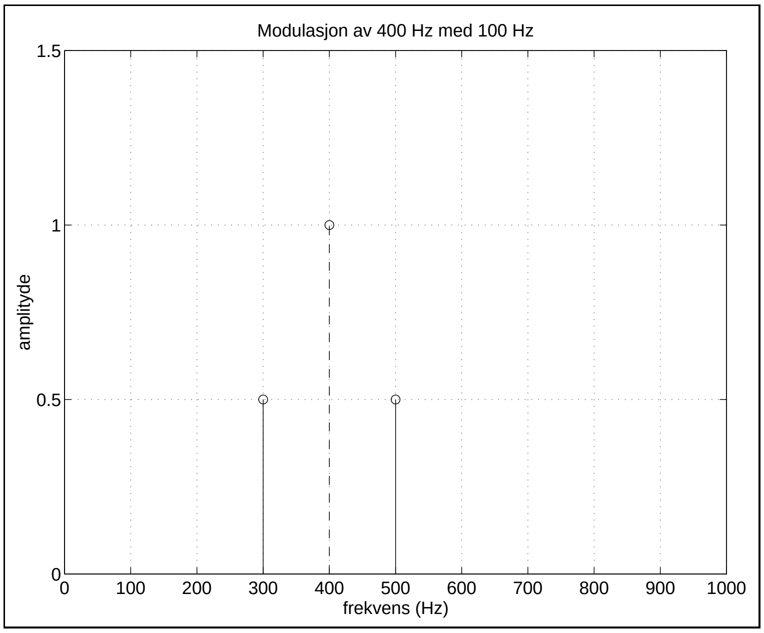
Ved å endre kontrollsignalet over tid, kan vi lage envelopes. Siden øret ikke kan oppfatte hurtige styrkevariasjoner, er det mulig å benytte en lav samplingfrekvens i envelopetabellen, kanskje under 1 kHz. I programmet C-Sound gjøres dette ofte for å spare beregningstid. Det kan da imidlertid ofte oppstå klikk. Den ideelle løsningen er kanskje å bruke en liten envelope-tabell, men å interpolere mellom samplene i kontrollsignalet.

Hvis vi modulerer med en sinustone, får vi *ringmodulasjon.* [1] Hvis vi modulerer med en sinustone med frekvens f_m, vil hver av deltonene i lydspekteret forsvinne. I stedet får vi to nye deltoner, med avstand $\pm f_m$ fra den opprinnelige deltonen og med halve amplityden. Disse deltonene kalles *sidebånd.* Som vi forklarte i heftet om akustikk, kan dette vises ved hjelp av formelen for multiplikasjon av to sinusoider:

$$cos(a)cos(b) = \frac{1}{2}[cos(a+b) + cos(a-b)].$$

Figuren under demonstrerer dette.

[1] Også kjent under navn som blanding, heterodyning, balansert modulasjon



Den opprinnelige frekvenskomponenten på 400 Hz er her utsatt for ringmodulering med modulatorfrekvens 100 Hz. Vi har da fått sidebånd på 300 Hz og 500 Hz.

Ringmodulasjon er en "klassisk" metode for lydomforming som ble mye brukt særlig på 50- og 60-tallet (Stockhausen, Nordheim etc.).

2.2 Miksing

Miksing av digital lyd skjer simpelthen ved at man adderer korresponderende sampleverdier. Dette krever at alle lydene er samplet med samme frekvens. Akkurat som i analog lyd må man passe på at summen holder seg innenfor maskinens dynamiske område, ellers får man klipping.

Når vi mikser vil vi gjerne fade ting inn og ut. Dette gjøres ved å legge på envelopes på hver lyd samtidig med addisjonen.

2.3 Kurveform-syntese

Den enkleste metode for lydsyntese er at brukeren simpelthen angir kurveformen direkte som et oscillogram. Kurven kan tegnes med et tegneverktøy på skjermen, eller genereres ved en angitt matematisk funksjon. Den vanligste metoden er å legge en periode av kurven inn i en tabell med opptil noen få tusen samples. Denne tabellen presses eller strekkes automatisk, avhengig av hvilken grunntonefrekvens vi ønsker, og så gjentas (loopes) denne perioden så lenge vi vil. Vi kan siden legge på envelopes eller behandle lyden ved andre omformingsmetoder.

Det er viktig at kurven ikke har altfor skarpe kanter. Dette vil kunne innebære frekvenskomponenter over Nyquist-frekvensen, slik at vi får aliasering.

Et annet lite problem ved kurveform-syntese (og generelt i digital lyd) er *interpolasjonsstøy*. Anta for eksempel at vi har en kurvetabell med 100 samples, og at samplingfrekvensen er 20 kHz. Dersom tabellen avspilles som den er, vil det ta $100 \cdot \frac{1}{20000}$ sekunder å spille en periode, hvilket gir en grunntonefrekvens på 200 Hz. Hvis vi nå ønsker en grunntonefrekvens på 300 Hz, må vi gå gjennom tabellen med en hastighet på 1.5 ganger det "normale". Måten å gjøre dette på, er å plukke ut hvert 1.5'te sample for avspilling. I praksis må da synteseprogrammet enten bare hente det nærmeste samplet i tabellen, eller man må interpolere. Slik interpolasjon er aldri perfekt, og dermed får vi støy.

Interpolasjonsstøy kan minimaliseres på to måter: Enten benytte en større kurveformtabell, eller benytte en bedre (og dermed mer beregningskrevende) interpolasjonsmetode. Vi får dermed en avveining mellom hukommelsesbruk, regnehastighet og lydkvalitet.

Kurveform-syntese er morsomt og lærerikt, fordi en lyd kan tegnes helt direkte. Med litt øvelse kan man også oppnå mange spennende effekter. Men øret oppfatter nå engang lyden i frekvensdomenet, slik at det ofte kan være vanskelig å forutsi hvordan en lyd som er angitt i tidsdomenet vil klinge. Dessuten kan det være kronglete å oppnå dynamikk i klangfarge.

Kurveform-syntese er en metode som har vært kommersielt populær, under navn som "PCM-syntese" (Casio) og "Waveform Synthesis" (Korg).

2.4 Additiv syntese

Additiv syntese går simpelthen ut på at brukeren angir spekteret direkte, enten statisk eller varierende over tid. Datamaskinens oppgave er så å omforme lyden fra frekvensdomenet til tidsdomenet. Fra Fourier's teorem følger det at enhver kurveform kan syntetiseres ved hjelp av additiv syntese. Kommunikasjonen med brukeren foregår dessuten i frekvensdomenet, som er den samme representasjonsform som øret benytter. Dette gjør additiv syntese til en meget kraftig metode, bortsett fra at antall parametre (amplityde og frekvens til hver av deltonene) kan virke litt overveldende. Ofte vil imidlertid disse parametrene ha sitt grunnlag i en foregående, maskinell spektralanalyse av en samplet lyd. Vi skal komme mer tilbake til slik *analyse-resyntese*.

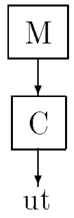
Additiv syntese kan være en nokså beregningskrevende sak, fordi man skal regne ut et stort antall sinusoider med varierende amplityde og mikse disse sammen. En alternativ og langt hurtigere beregningsmetode er å bruke invers Fast Fourier-transformasjon (IFFT). Dette kommer vi også tilbake til.

Det finnes et antall kommersielle synthesizere som benytter additiv syntese (f.eks. fra Kawai).

2.5 FM-syntese

FM-syntese er kanskje den mest populære digitale syntesemetoden, ikke minst på grunn av Yamahas DX7-serie. FM er en indirekte og lite intuitiv teknikk. Det krever øvelse å kunne bestemme synteseparametrene riktig for å få en ønsket lyd. Grunnen til denne metodens popularitet er først og fremst at den kan produsere forholdsvis komplekse klanger uten at datamaskinen behøver å slite for mye. Det var amerikaneren *John Chowning* som først pekte på FM-syntese som musikalsk verktøy.

Som en introduksjon til FM kan vi betrakte et oppsett som lager sinustoner med vibrato. Vi benytter to sinusoscillatorer: En oscillator C som lager selve lyden, og en lavfrekvent oscillator (LFO) M som styrer frekvensen til C. Dette kan tegnes slik:



I FM-sjargong kalles M for *modulator*, og C for *carrier*. [2] Frekvensen som M svinger med kalles modulatorfrekvensen, og gjennomsnittsfrekvensen til C kalles *carrierfrekvensen*. Amplityden til M heter *modulasjonsindeks*, og benevnes her med bokstaven I. Amplityden til C spiller bare rolle for lydstyrken. I dette systemet har vi dermed tre interessante parametre som kan endres: Modulatorfrekvens, carrierfrekvens og modulasjonsindeks.

M svinger her med en lav frekvens, og vi får en vanlig sinus med vibrato. Dersom vi imidlertid øker frekvensen til modulator opp i det hørbare området, vil vi få kraftig endring av klangfargen. Hvilken klangfarge vi får, avhenger av forholdet mellom frekvensen til C og frekvensen til M (såkalt C:M ratio) og av modulasjonsindeksen.

Vi kan sette opp følgende "regler" for hvordan spekteret til ovenstående system er bygget opp:

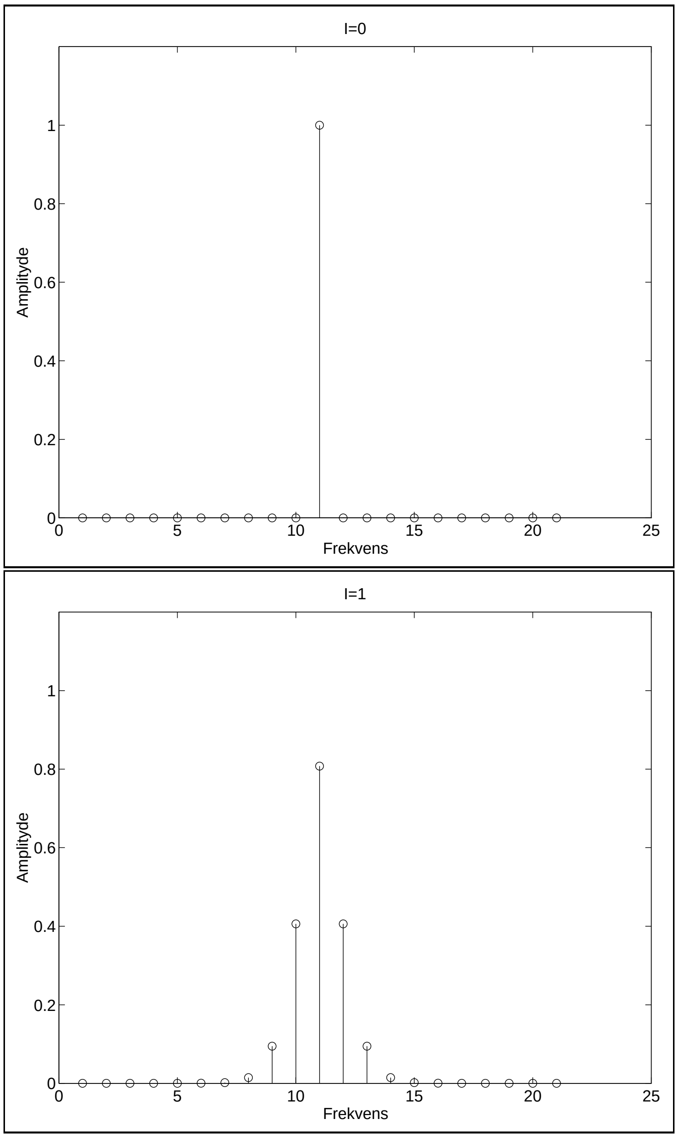
- Carrier vil være tilstede som en deltone i spekteret
- På hver side av carrierfrekvensen vil det oppstå et stort antall nye frekvenskomponenter (sidebånd)
- Avstanden fra carrierfrekvensen til de to nærmeste sidebånd og mellom hvert av sidebåndene tilsvarer modulatorfrekvensen [3]
- Amplityden til carrier og sidebåndene avhenger av modulasjonsindeksen. Jo større I, jo svakere carrierfrekvens og jo sterkere og flere sidebånd. Vi kan si at det stjeles energi fra carrier som fordeles utover på sidene. Dette kan tallfestes ved hjelp av såkalte *Bessel-polynomer*.
- Det skjer nesten alltid at de nederste sidebåndene får en frekvens som er mindre enn 0. Disse reflekteres da oppover i spekteret igjen, og blander seg med de andre komponentene.

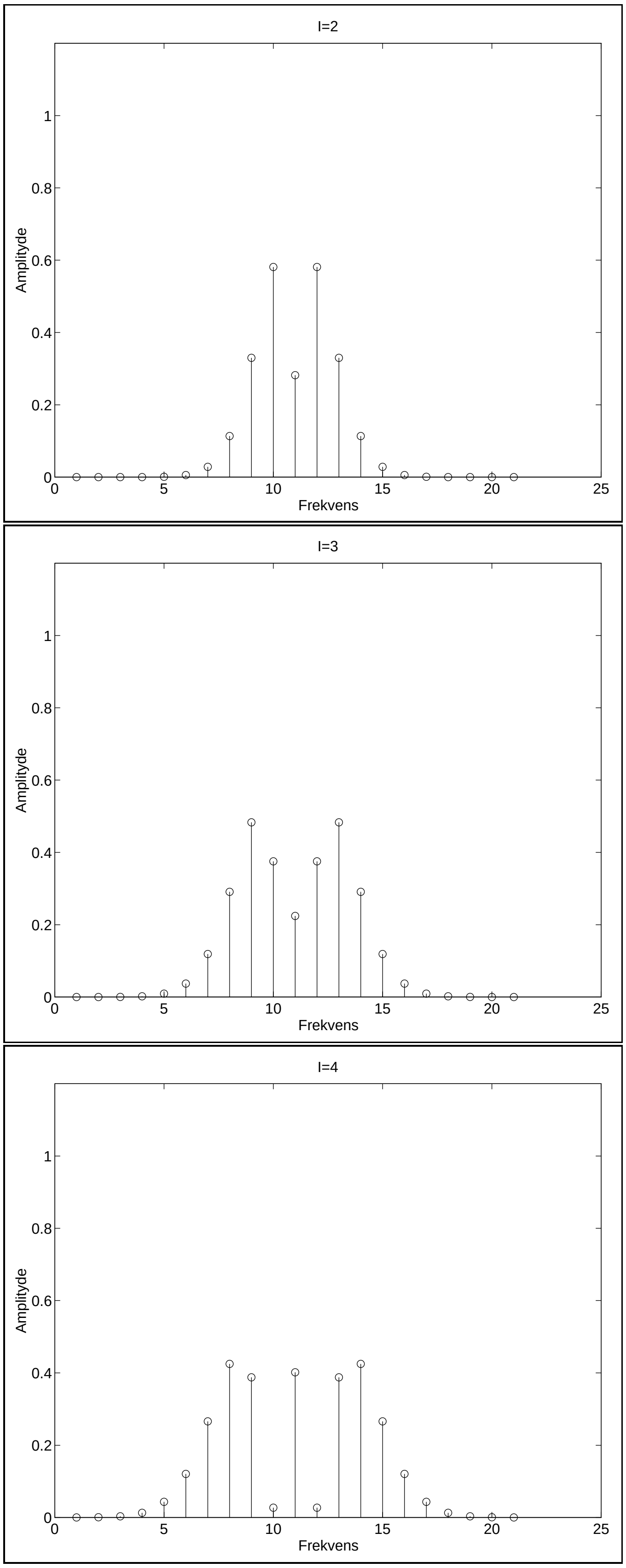
[2] Denne ordbruken er tatt fra radioteknikken

[3] Bemerk analogen til ringmodulasjon, men der hadde vi bare to sidebånd

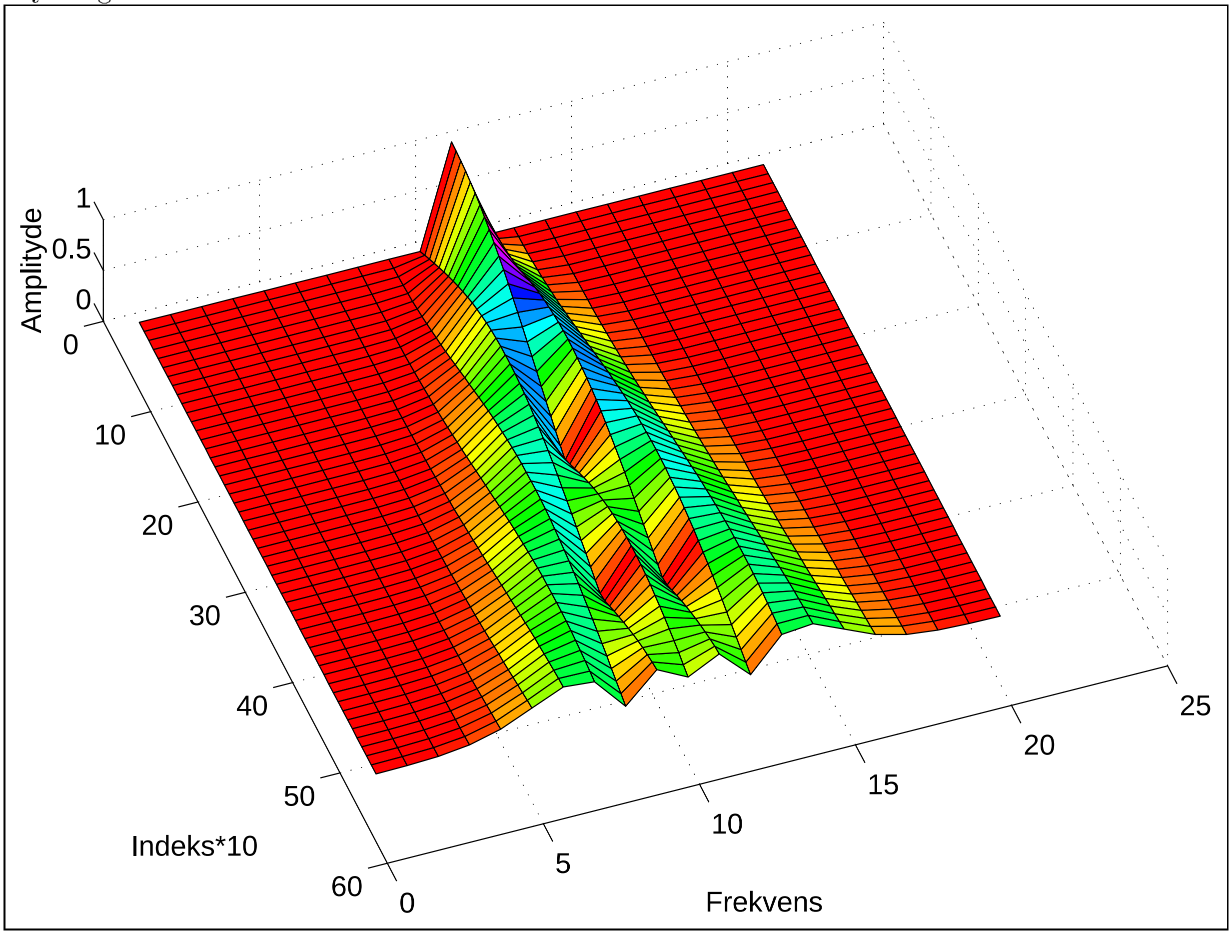
- Disse refleksjonene forklarer hvorfor heltallige C:M-forhold gir harmoniske spektre (refleksjonene treffer da akkurat på de opprinnelige deltonene), mens ikke-heltallige C:M-forhold gjerne gir inharmoniske spektre (refleksjonene treffer da innimellom de opprinnelige deltonene).

I figurene under kan vi se hvordan spekteret blir rikere ettersom vi øker modulasjonsindeksen. Carrierfrekvensen er her så stor i forhold til modulatorfrekvensen at vi ikke får refleksjoner.

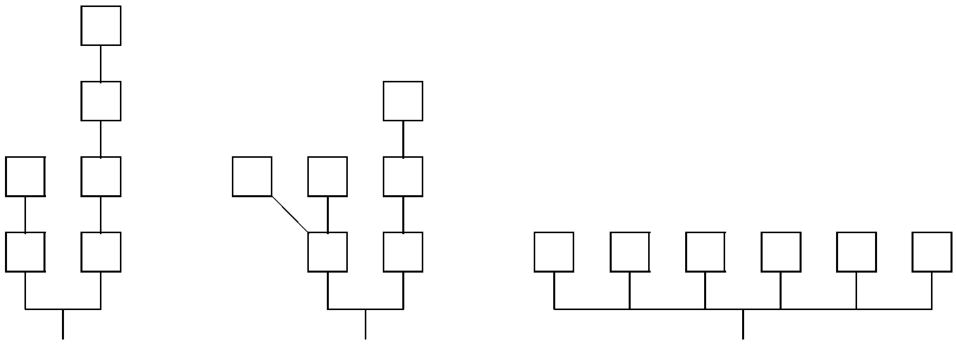



I=2
Amplityde
Frekvens
I=3
Amplityde
Frekvens
I=4
Amplityde
Frekvens

Det er fristende å vise dette i tre dimensjoner, selv om situasjonen kanskje blir litt utydelig:



Foreløpig har vi betraktet et system med bare to oscillatorer: En carrier og en modulator. Parametrene er dessuten statiske. I FM-syntese pleier vi imidlertid å bruke flere enn to oscillatorer. For å oppnå dynamikk i klangfarge, legger vi envelopes på modulatorene, og for å få dynamikk i styrke, legger vi envelopes på carrierne. Av dette følger en generell byggeblokk som kan brukes både som carrier og modulator, nemlig en oscillator med envelope på utgangen og med styrbar frekvens. En slik byggeblokk kalles en *operator*. DX7 har 6 slike operatorer, som kan kobles sammen på 32 forskjellige måter. En slik sammenkoblingsmåte (patch) kalles en *algoritme*. Her en noen eksempler på FM-algoritmer:



Algoritmen lengst til høyre er egentlig ikke FM-syntese, men en enkel additiv syntese med 6 deltoner.

2.6 Ulineær kurveforming

Denne metoden heter på engelsk *Non-linear Distortion*, på fransk *Distortion Non-Lineaire*, forkortet DNL. Poenget er å kjøre lyd (sinustoner eller naturlig lyd) gjennom en ulineær overføringsfunksjon. Denne funksjonen kan f.eks. implementeres som en tabell der input-sampleverdien brukes som en indeks som peker ut en verdi i tabellen. Denne verdien blir output-verdi.

Overføringsfunksjonen kan tegnes på en dataskjerm, eller den kan beregnes matematisk. Enhver funksjon kan legges inn i tabellen. Her er noen eksempler (du finner flere i heftet om matematikk):

$$f(x) = x$$

(identitetsfunksjonen) endrer ikke lyden i det hele tatt, fordi output-samplene $f(x)$ er de samme som input-samplene x.

$$f(x) = 2x$$

vil doble amplityden. Dette gir en forsterking på 6 dB. Disse to funksjonene er begge lineære; funksjonsgrafen er en rett linje. Dette betyr at klangfargen beholdes.

$$f(x) = abs(x)$$

der $abs(x)$ er absoluttverdien av x (fortegnet kastes) er en ulineær funksjon som endrer lydens klangfarge og gir en fuzz-aktig effekt.

Av spesiell interesse er en type overføringsfunksjoner som kalles *Chebychev-polynomer*. Disse polynomene har den interessante egenskap at dersom vi sender en sinustone gjennom dem, får vi en harmonisk ut. Et polynom av andre grad gir andre harmoniske, tredje grad gir tredje harmoniske osv. Ved å legge sammen flere polynomer, får vi et nytt polynom som kan produsere et helt spektrum når vi sender en sinustone gjennom det. Vi har da full kontroll over deltonenes amplityder. Dette forutsetter imidlertid at input-sinusen har amplityde 1. Ved mindre amplityder får vi et fattigere spektrum ut. På denne måten vil klangen endre seg (bli rikere og lysere) med større amplityde. En liknende oppførsel er kjent fra de fleste musikkinstrumenter.

Det bør nevnes at både kurveform-syntese og FM-syntese kan betraktes som spesialtilfeller av ulineær kurveforming. Dette er nok imidlertid mer av teoretisk enn av praktisk interesse.

2.7 Amplityde-følging

RMS-analyse (Root-Mean-Square) er den klassiske måte å måle signalstyrke på. En signalstyrkemåler kan brukes til å trekke ut amplityde-envelopen til en lyd. Denne envelopen kan så endres eller anvendes på andre lyder. Dette siste er et enkelt eksempel på *kryss-syntese*, der man trekker ut informasjon fra en lyd og anvender på en annen.

Ved RMS-analyse kvadrerer vi først hvert sample. Dette gjør at vi bare får å gjøre med positive verdier i det følgende. Deretter integrerer vi (tar gjennomsnittet) over et visst tidsrom. Bemerk at dersom vi ikke hadde kvadrert til å begynne med ville enhver sinusoid integrere til null, og resultatet ville vært verdiløst. Til slutt trekker vi ut kvadratroten for å oppveie kvadreringen. Vi beregner altså kort sagt *roten av gjennomsnittet av kvadratet*, derav navnet RMS.

Det interessante spørsmålet her er hvordan og over hvilke tidsrom vi skal ta gjennomsnittet. Det finnes flere forskjellige måter å produsere et gjennomsnitt på:

- *Langtids-gjennomsnitt*, der vi setter integrasjonstiden T lik hele obervasjonstiden. Vi får da ut ett enkelt tall, som er den gjennomsnittlige amplityden for hele lyden.

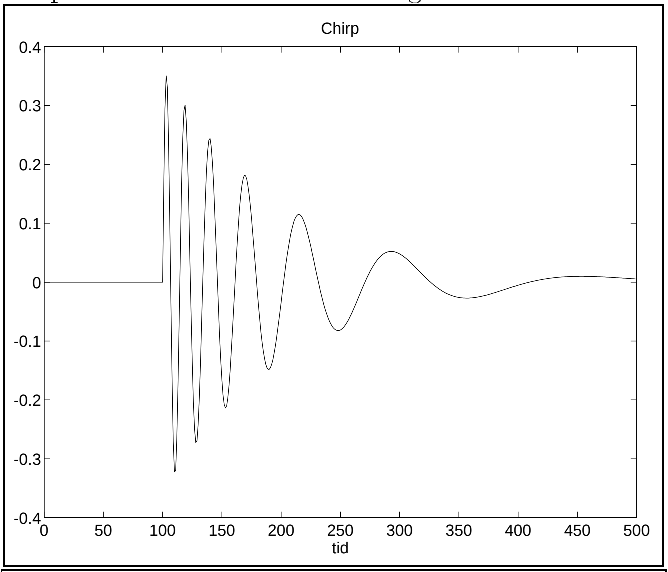
- *Stegvis gjennomsnitt*, der vi tar gjennomsnittet over en lyd-bit som er T sekunder lang. Deretter tar vi gjennomsnittet over en ny bit som starter ved slutten av den forrige, etc. Vi får dermed ut en amplityde-verdi for hver bit. Hvis $T = 0.1$ sek får vi ut 10 RMS-verdier i sekundet.

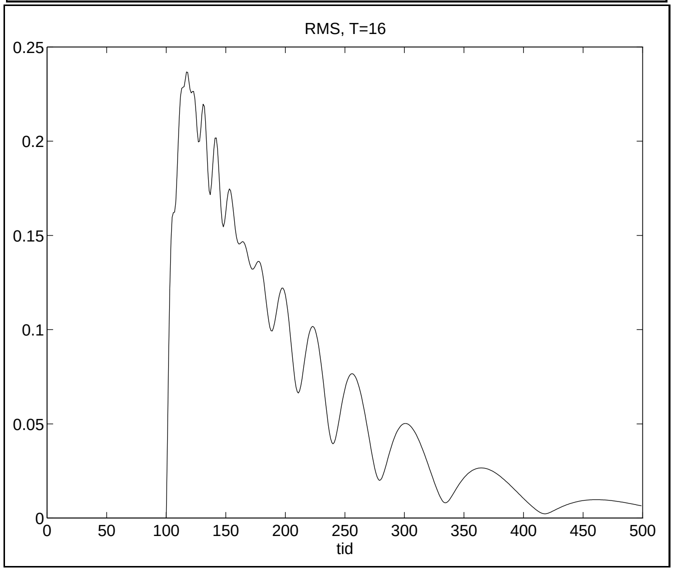
- *Løpende gjennomsnitt* er en mer korrekt metode, der vi hele tiden tar gjennomsnittet over de T siste sekundene. Vi får da like mange RMS-samples ut som lyd-samples inn, men siden innholdet av høye frekvenser vil være lite kan vi nokså trygt sample RMS-verdiene med en lavere frekvens uten at vi får aliasering.

- *Vektet gjennomsnitt* tillegger samplene forskjellig vekt i gjennomsnittet. Som oftest blir "gamle" samples dempet i forhold til "nye". Lavpassfiltre som demper på en meget lav frekvens (størrelsesorden 20 Hz) kan brukes til en slik gjennomsnitting. Envelope-følgeren som vi vanligvis bruker på IRCAM-maskinen fungerer på denne måten.

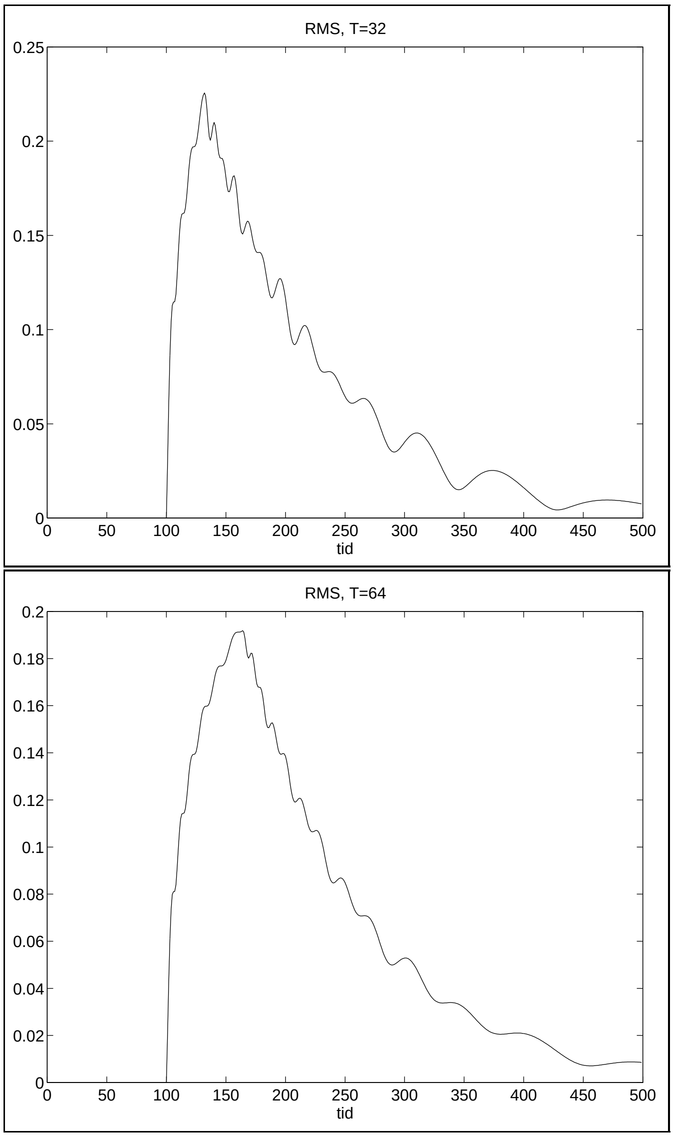
Integrasjonstiden T bestemmer hvor nøye envelopen skal tracke lyden. Vi ønsker ikke at trackingen skal være *for* nøyaktig, for da kan vi risikere å begynne å følge selve kurveformen ved lavfrekvente signaler. På den annen side må T være såpass liten at transienter ikke dempes for mye. Dette er en vanskelig balansegang, og T bør helst settes manuelt ettersom hvaslags lyd man har med å gjøre.

Figurene under viser tre eksempler på RMS-måling med løpende integrasjon, med ulike T. Testlyden er en sinustone med en skarp attack. Tonen dør så ut samtidig med at frekvensen senkes. [4] Legg merke til hvordan transientresponsen, som observeres i begynnelsen av signalet, blir dårligere ettersom vi øker T. Men samtidig dempes selve signalet bedre ved de lave frekvensene på slutten. $T = 32$ er muligens et brukbart kompromiss.

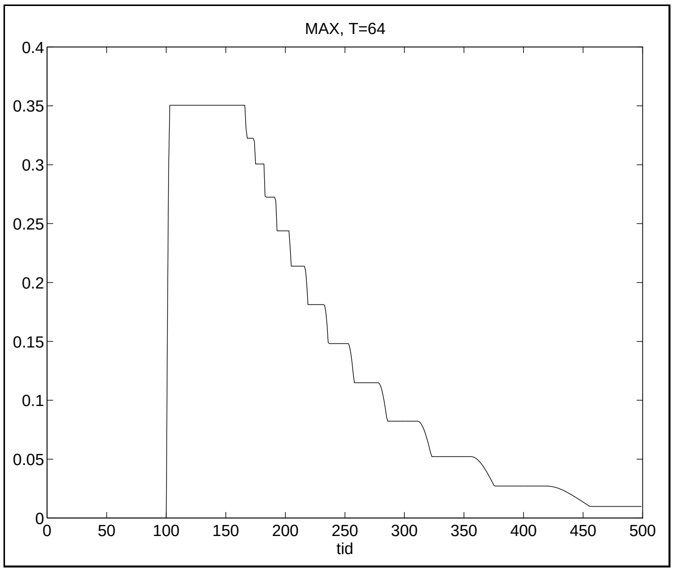




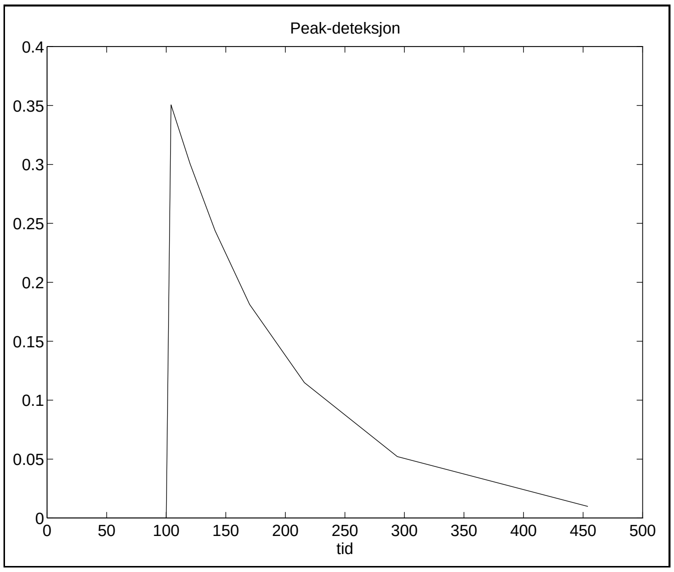
[4] Dette er et mye brukt test-signal, og kalles en *chirp*



Vi kan prøve etpar andre metoder også. Hvis vi velger ut den *maksimale* verdien i de siste 64 samples, får vi dette:



En annen teknikk er *peak-deteksjon*, det vil si at vi forsøker å finne toppene i kurven og trekker linjer mellom disse:



Akkurat for denne lyden fikk vi her et glimrende resultat, men generelt vil denne metoden gå i vasken når vi begynner å få inn overtoner som lager "småtopper" mellom hovedtoppene. Det finnes imidlertid endel triks for å forsøke å overse disse.

2.8 Pitch-følging

En annen viktig analyse-teknikk er *pitch-følging* eller F_0-*analyse*, der vi forsøker å plukke ut grunntonefrekvensen i et signal. Dette forutsetter naturligvis at vi har å gjøre med en monofon lyd med en veldefinert grunntone.

Å finne frekvensen til en grunntone er en meget komplisert affære, og det er ikke klart hvordan øret og hjernen løser denne oppgaven. Et stort antall metoder er pønsket ut for pitch-følging, men ingen av dem er ufeilbarlige. En vanlig feil er at analysatoren plukker ut frekvensen til en overtone, slik at vi kan få f.eks. en pitch som ligger en oktav for høyt.

En klassisk teknikk er peak-deteksjon, der vi teller antall bølgetopper i sekundet. For å unngå å telle små-topper som stammer fra overtoner, kan vi gjøre en kraftig lavpassfiltrering først. Dette lavpassfilteret bør følge grunntonefrekvensen gjennom en feedback- mekanisme.

En liknende metode er å telle nullgjennomganger, dvs. de punktene der kurven går fra negative til positive verdier. Igjen vil overtoner kunne lage problemer, og vi bør lavpassfiltrere signalet først.

En av de smartere metodene går ut på å benytte et *kamfilter* som slipper gjennom en viss grunntonefrekvens og harmoniske av denne, men demper alle andre frekvenser. Et slikt filter er lett å lage. Grunntonefrekvensen varieres så inntil vi finner et maksimum i signalstyrken ut fra filteret. Vi kan da regne med at toppene i filterkarakteristikken har truffet deltonene i det opprinnelige signalet. Ettersom pitch varierer, passer vi på å oppdatere filteret slik at signalstyrken forblir størst mulig. En slik algoritme kan bli lurt ved at den stabiliserer seg på en *subharmonisk*. Hvis vi har en grunntonefrekvens på 300 Hz, kan vi risikere at analysatoren finner et maksimum for 100 Hz, fordi topp nr. 3, 6, 9 etc. i kamfilteret treffer deltonene på 300 Hz, 600 Hz, 900 Hz etc. Man bør derfor forsøke å finne den *største* frekvensen som gir maksimum.

Det finnes også metoder som arbeider i frekvensdomenet, ved at man gjør en spektralanalyse og finner toppene i spekteret. Grunntonefrekvensen bestemmes så som største felles faktor av disse toppene.

Kapittel 3

Fysisk modellering

Fysikerne begynner etterhvert å få ganske god oversikt over hvilke prosesser som foregår i vibrerende musikkinstrumenter. Vi kan lage matematiske modeller for dette, og implementere dem på datamaskinen. Vi *simulerer* således de fysiske prosessene i musikkinstrumentet, og kan på den måten produsere lyd.

Denne syntesemetodikken har mange tiltalende sider. For det første kan man produsere lyd som låter svært naturlig og ekte. For det andre kan en musiker kommunisere med datamaskinen gjennom en fysisk begrepsverden, og f.eks. bygge opp et musikkinstrument på skjermen ved hjelp av objekter som strenger, resonanskasser, membraner, buer, hammere, plektre, rør etc. Man kan benytte hvilke byggematerialer man vil, og lage instrumenter av alle kategorier og størrelser. Det er også mulig å lage gradvise overganger fra ett instrument til et annet ("morfing"). Syntetiske instrumenter basert på fysisk modellering vil også reagere på en ordentlig måte på forskjellige spilleteknikker. Hvis man f.eks. presser hardt på en virtuell fiolinbue, kan man forvente at modellen gir en styggere lyd på samme måte som en ekte fiolin gjør det.

Mange teknikker for fysisk modellering er svært beregningskrevende. Ofte dreier det seg om å løse såkalte *partielle differensiallikninger* numerisk. *Bølgelikningen*, som beskriver hvordan bølger forplanter seg, og *Navier-Stokes-likningen*, som beskriver turbulent luftstrømning, er to slike likninger som dukker opp i musikalske anvendelser. Å løse partielle differensiallikninger koker gjerne ned til å løse lineære likningssystemer med mange tusen likninger og tilsvarende antall ukjente. Dette tar tid.

I de senere år er det imidlertid kommet både maskiner som er raske nok til å gjøre slike ting på en rimelig tid, og nye, mer effektive algoritmer. Fysisk modellering står nå antagelig foran et gjennombrudd både i eksperimentell og kommersiell musikk.

3.1 En enkel fiolin-modell

La oss se på et enkelt eksempel hvor vi prøver å etterlikne en fiolinlyd ut fra fysiske betraktninger. I heftet om akustikk så vi at vi kan anvende en eksitator-resonator-modell på fiolinen. Eksitator består av en sagtann-liknende kurve, som stammer fra "slip and grip"-interaksjonen mellom bue og streng. En slik sagtann kan vi enkelt generere i en analog synthesizer eller i datamaskinen.

Resonator består av fiolinkassen og andre mekaniske elementer som filtrerer sagtann-kurven fra strengen. Frekvensresponsen til fiolinkassen kan måles i laboratoriet, eller vi kan prøve å regne den ut på grunnlag av kassens form (det er ikke lett!). Vi lager oss så et filter med tilsvarende respons, og kjører sagtannkurven gjennom dette.

Dette er en enkel modell, der vi har utelatt et utall av subtile fysiske effekter. For eksempel vil ikke eksitator gi en ren sagtann-kurve, men en komplisert kurve som bl.a. avhenger av buetrykket.

Et annet problem er at da vi konstruerte resonatoren, tok vi utgangspunkt i fysiske målinger på en virkelig fiolin. Dette gjør modellen lite fleksibel.

3.2 Karplus-Strong-algoritmen

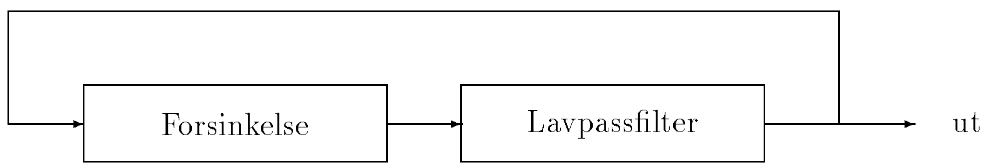
"The Karplus-Strong Plucked String Algorithm" er en spesialisert metode for å syntetisere gitar-liknende lyder. Denne algoritmen er kanskje lite fleksibel, men den gir glitrende god lyd og er ekstremt beregningseffektiv.

La oss si at vi sitter på et fast sted på en gitarstreng og observerer hva som farer forbi. På dette punktet har vi også plassert en mikrofon hvor vi plukker opp strengens utslag.

Når strengen klimpres på et punkt et stykke unna oss, vil strengens deformasjon forplante seg utover til begge sider. Vi kan regne med at denne deformasjonen har en nokså tilfeldig form, med et rikt spektrum. På veien mot oss vil imidlertid de høyeste frekvenskomponentene dempes mer enn de laveste. Dette har å gjøre med hvordan friksjonen behandler ulike frekvenser forskjellig. Strengen fungerer altså som et *lavpassfilter*.

Denne noe lavpassfiltrerte kurveformen farer forbi oss, og blir samtidig plukket opp av mikrofonen. Bølgen farer så videre mot enden av strengen, blir reflektert, lavpassfiltrert en gang til på veien, og kommer tilbake til oss. Slik pendler bølgebevegelsen fram og tilbake, samtidig med at den filtreres stadig mer. Bevegelsen dør da etterhvert ut, og de høyeste frekvensene forsvinner først.

La oss prøve å lage en modell for dette:



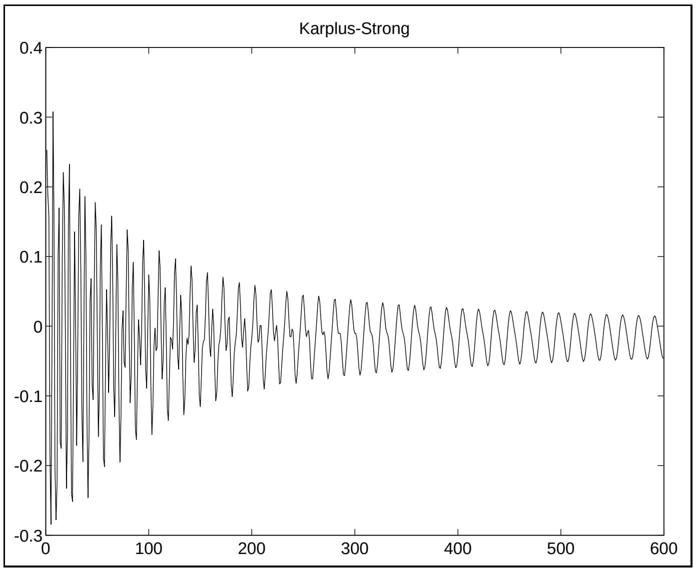
Forsøk å se sammenhengen mellom forklaringen over og hvordan signalet går i modellen!

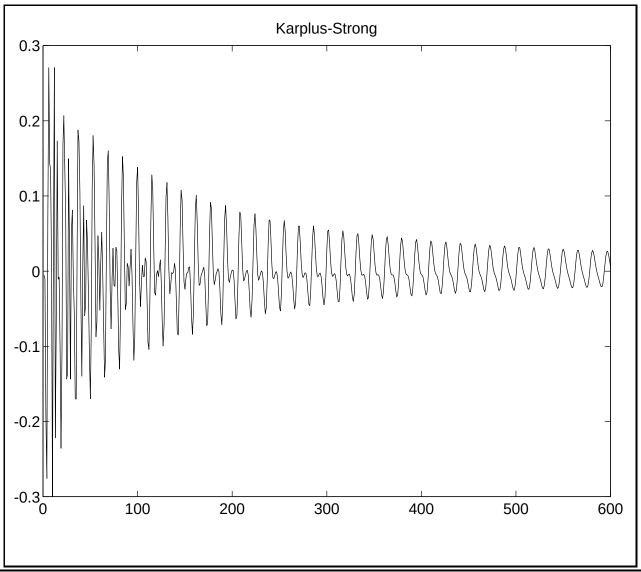
Forsinkelsen implementeres som en digital delay. Grunntonefrekvensen har å gjøre med lengden på strengen, og dermed hvor lang tid det tar for bølgen å komme tilbake til vår tapping. Lengden på forsinkelsen vil dermed bestemme grunntonefrekvensen: 1 millisek vil gi 1000 Hz, 2 millisek 500 Hz etc.

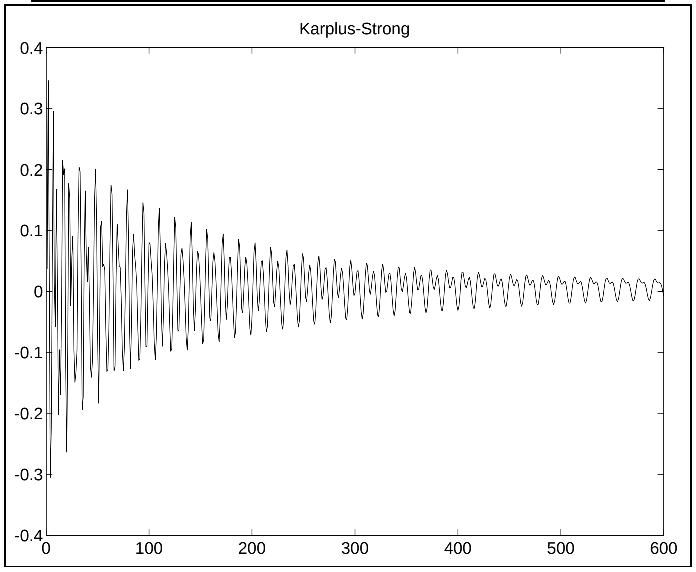
Lavpassfilteret kan være av enkleste type. Det er mer enn godt nok å simpelthen ta gjennomsnittet av nåværende og foregående sample. Av dette ser vi at Karplus-Strong-algoritmen nesten ikke krever regnekapasitet i det hele tatt.

Vi ser også at KS-algoritmen i sin enkleste form kan implementeres ved hjelp av en vanlig digitaldelay (effektboks) som er innstilt på en meget liten forsinkelsestid, et lavpassfilter (f.eks. tonekontrollen på en mikser) og en feedback.

Vi starter en note ved å fylle digitaldelayen med en tilfeldig sekvens (hvit støy). Tilfeldigheten gjør at vi får en noe annerledes klang for hvert anslag, og dette øker naturligheten. Denne snutten vil så fare rundt i systemet, samtidig med at den lavpassfiltreres igjen og igjen. Vi ender etterhvert opp med bare grunntonen (en sinus), men den forsvinner også til slutt. Her er noen eksempler, der vi har brukt identiske parametre hver gang. Tilfeldigheten lager likevel variasjon i resultatet:







Ved å endre gain i feedback-løkken, kan vi styre hvor hurtig lyden skal dø ut. Vi kan si at vi da regulerer friksjonen i strengen.

Karplus-Strong-algoritmen kan utvides på mange måter. Vi kan benytte forskjellige slags filtre i feedback-løkken, filtrere sluttresultatet for å simulere gitarkasse, lage el-gitar-effekter som feedback og fuzz osv. En mykere attack kan oppnås ved å lavpassfiltrere hvitstøyen en gang før selve syntesen starter. Det går også an å putte naturlig lyd inn istedenfor hvit støy (dette er gjort i "Smalltalk" av Paul Lansky).

I det hele tatt er dette en morsom metode å arbeide med. En god demonstrasjon av Karplus-Strong-algoritmen er stykket "Silicon Valley Breakdown" av David Jaffe. Anders Vinjar i Norge har laget musikk som benytter utvidelser av Karplus-Strong-algoritmen. En implementasjon av Karplus-Strong finnes bl.a. i C-Sound.

3.3 Waveguides

Karplus-Strong-algoritmen er et spesialtilfelle av en mye større klasse av syntesemetoder basert på fysisk modellering, nemlig syntese ved *waveguides*. Dette er en helt ny metodikk, som er utviklet av Julius O. Smith III ved CCRMA, Stanford. Som i Karplus-Strong- algoritmen, forsøker man å sette opp digitale forsinkelser som etterlikner mediet som lydbølgene forplanter seg i. Langs disse "bølge-lederne" kan man sette inn elementer som modifiserer signalet på sin vei, f.eks. lavpassfiltre. På den måten kan man løse bølgelikningen på en effektiv og fleksibel måte.

Yamaha har nå satt i produksjon en synthesizer som benytter denne metoden. Den har bare to stemmer og koster 6000 dollar, men vi må regne med at dette bare er begynnelsen på en viktig kommersiell teknologi.

Ved CCRMA har man også utviklet et program for syntese av tale og sang, kalt SPASM, som bruker waveguides. Kvaliteten og fleksibiliteten til dette systemet skal være meget god.

3.4 Modal syntese

Det går også an å tenke fysisk modellering mer i frekvensdomenet: Hvilke frekvenser vil kunne finnes i det fysiske systemet, og hvordan vil de utvikle seg? Siden vi da ser på egenskapene til de ulike svingningsmodi som er tilstede, kan vi kalle dette for *modal syntese*.

Programmet *Mosaic*, som delvis er utviklet ved IRCAM, benytter modal syntese i et system som lar brukeren bygge opp sine egne instrumenter ved hjelp av forhåndsdefinerte virtuelle fysiske objekter.

I Norge har *Terje Syversen* utviklet teknikker og programvare for modal syntese. Han har også planlagt hardware som skal kunne utføre dette i sann tid. [1]

[1] Terje Syversen: Syntese av Musikalsk Lyd, hovedoppgave ved Institutt for Fysikk, Universitetet i Oslo

Kapittel 4

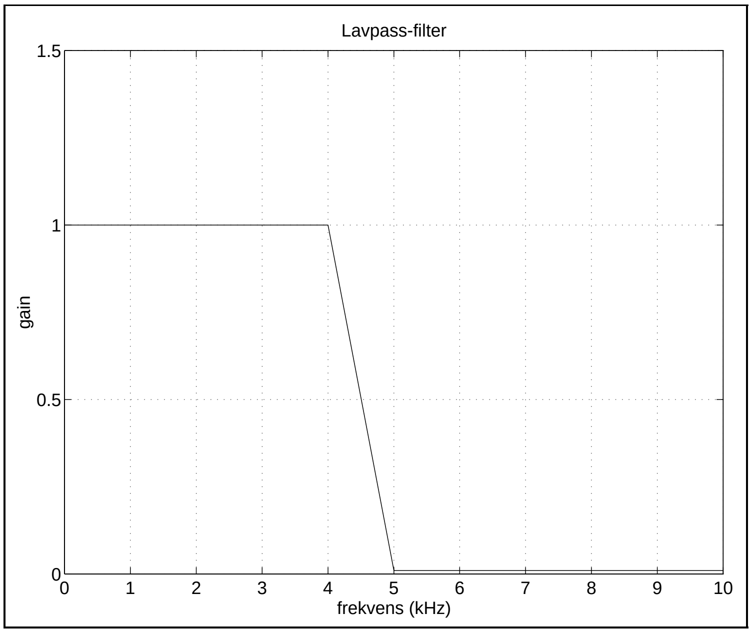
Lydbehandling og syntese i frekvensdomenet

Vi har hittil sett mest på lydbehandlingsteknikker som først og fremst betraktes i tidsdomenet. I dette kapittelet skal vi se på metoder der vi behandler lydens spektrum, slik at vi beveger oss i frekvensdomenet. Dette er en noe løs klassifisering, siden enhver operasjon i tidsdomenet gir en endring i frekvensdomenet og omvendt. Det er, som vi har sett, bare snakk om to forskjellige måter å representere lydsignalet på.

4.1 Generelt om filtre

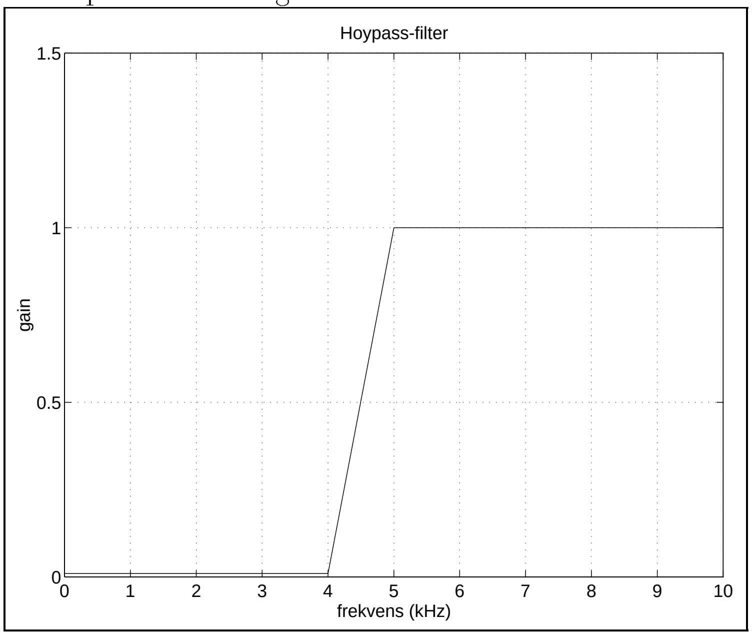
Et *filter* er en innretning som endrer et spektrum på en slik måte at det ikke kommer til nye frekvenskomponenter. For å være litt mer konkret, kan vi si at et filter generelt *demper* visse frekvenser og *forsterker* andre. Tradisjonelt betraktes følgende klasser av filterkarakteristikker som særlig viktige:

- *Lavpass*, der alle frekvenser under en viss frekvens (*cut-off* eller *klippefrekvens*) beholdes, mens høyere frekvenskomponenter dempes mest mulig:



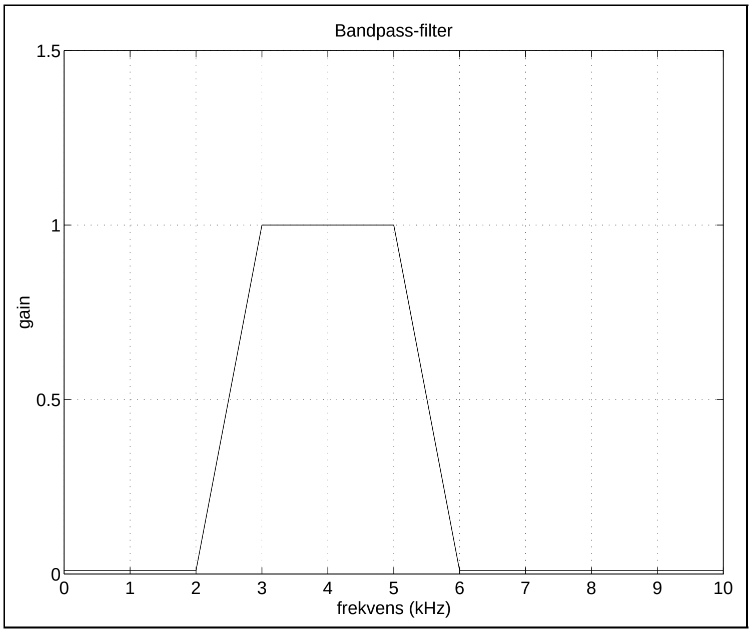
Et lavpassfilter kan dermed f.eks. brukes til å dempe støy på høye frekvenser. Diskant-knappen på stereoanlegget kontrollerer et lavpassfilter.

- *Høypass*, der alle frekvenser over klippefrekvensen beholdes, mens lavere frekvens-komponenter dempes mest mulig:



Bass-knappen på stereoanlegget kontrollerer et høypassfilter. Et høypassfilter og et lavpassfilter brukes i to-veis høyttalere for å styre lyd til henholdsvis diskant- og bass-elementet.

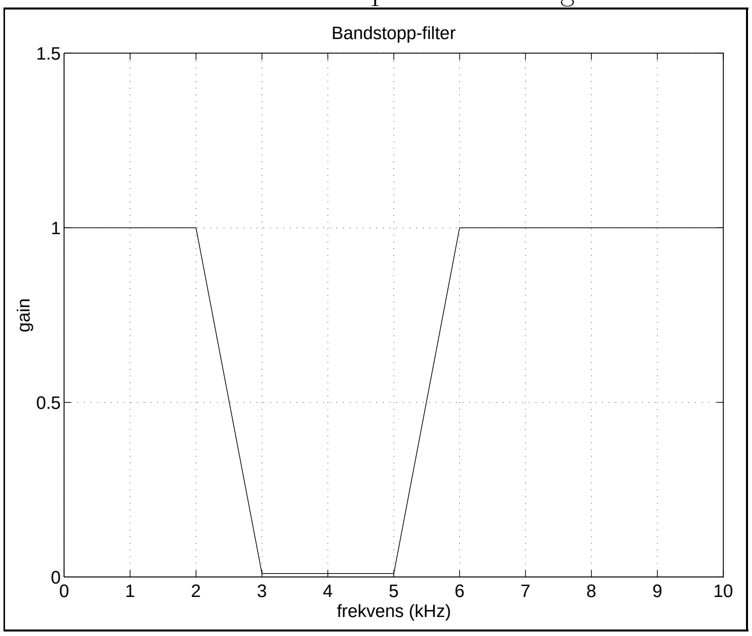
- *Båndpass*, der alle frekvenser innenfor et visst bånd beholdes, mens frekvenskomponenter utenfor båndet dempes mest mulig:



Et båndpassfilter kan karakteriseres ved en *senterfrekvens* (her 4 kHz) og en *båndbredde*. Ingeniører snakker gjerne om *Q-faktor* (godhetsfaktor), som er definert som senterfrekvens delt på båndbredde. Stor Q betyr dermed liten båndbredde.

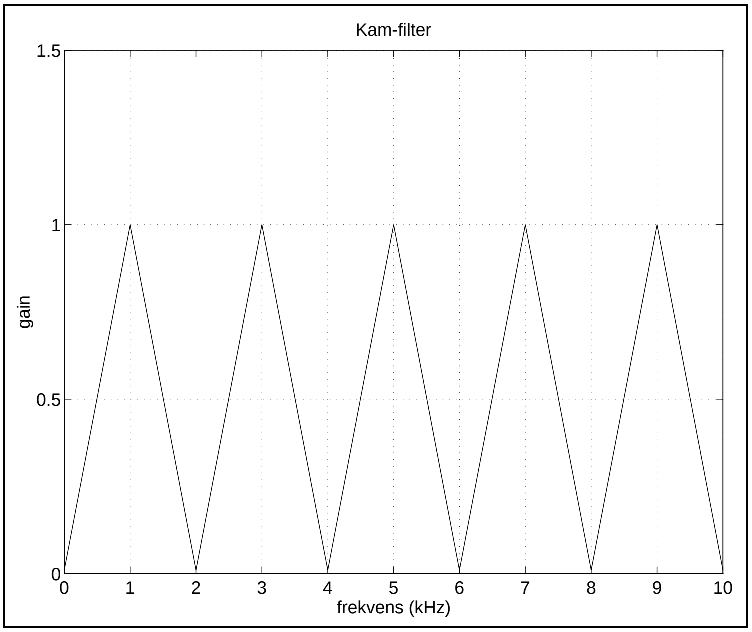
Et båndpassfilter med liten båndbredde kan "sveipes" over en lyd slik at bare en og en deltone slipper igjennom filteret av gangen.

- *Båndstopp* eller *notch*, der alle frekvenser utenfor et visst bånd beholdes, mens frekvenskomponenter innenfor båndet dempes mest mulig:



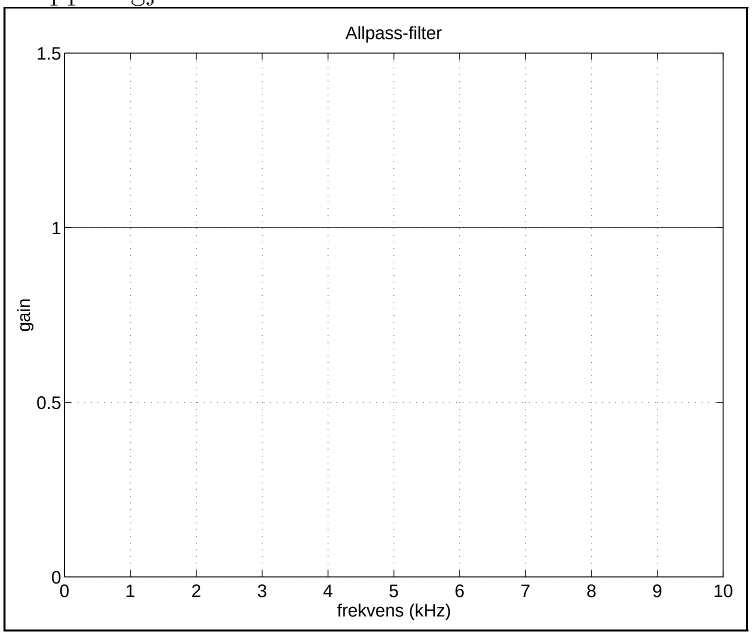
Et båndstoppfilter med liten båndbredde og senterfrekvens 50 Hz er fint til å fjerne nettbrum med.

- *Kamfilter*, som vekselvis stanser og stopper frekvensene med jevne mellomrom oppover i spekteret:



Et kamfilter er fint til å slippe igjennom en grunntone med overtoner, men dempe alt annet. Vi så en anvendelse for dette i avsnittet om pitch-følging.

- *Allpass*, som slipper igjennom alle frekvenser uhindret:



Hva i all verden er vitsen med det? Et allpassfilter gjør kanskje ingenting med *styrken* til de ulike deltonene, men det kan ha stor effekt på *fasen*. Dette betyr at man kan konstruere allpassfiltre som slipper lyden uhindret igjennom, bortsett fra at ulike frekvenser forsinkes ulikt og derfor kommer ut til forskjellig tid. Dette kalles *dispersjon*, og er analogt til spredningen av ulike frekvenser i lys gjennom et prisme.

Det er teoretisk umulig å konstruere et filter hvor flankene er helt loddrette. Det vil alltid finnes et overgangsområde mellom passbåndet og stoppbåndet. Generelt må et filter gjøres mer komplisert jo steilere flanker vi ønsker. De enkleste filtrene (av *første orden*) er lavpass- og høypassfiltre som demper 6 dB pr. oktav. Mer kompliserte filtre av annen orden kan dempe 12 dB pr. oktav. Svære filtre, ofte kalt *brick-wall-filtre*, kan dempe kanskje oppimot 100 dB pr. oktav, men da begynner man å få problemer med bl.a. stor forsinkelse gjennom filteret.

Vi kan lage mer kompliserte karakteristikker og brattere flanker ved å koble grunntypene over i serie (da vil karakteristikkene multipliseres) eller parallell (da vil karakteristikkene adderes).

4.2 Digital filtrering ved differenslikninger

Filtrering kan implementeres meget effektivt på en datamaskin ved hjelp av *differenslikninger*. En slik likning angir ett nytt utgangs-sample som en vektet sum av tidligere inngangs-samples og utgangs-samples. Hvis x er inngangs-samples og y er utgangs-samples, kan vi skrive

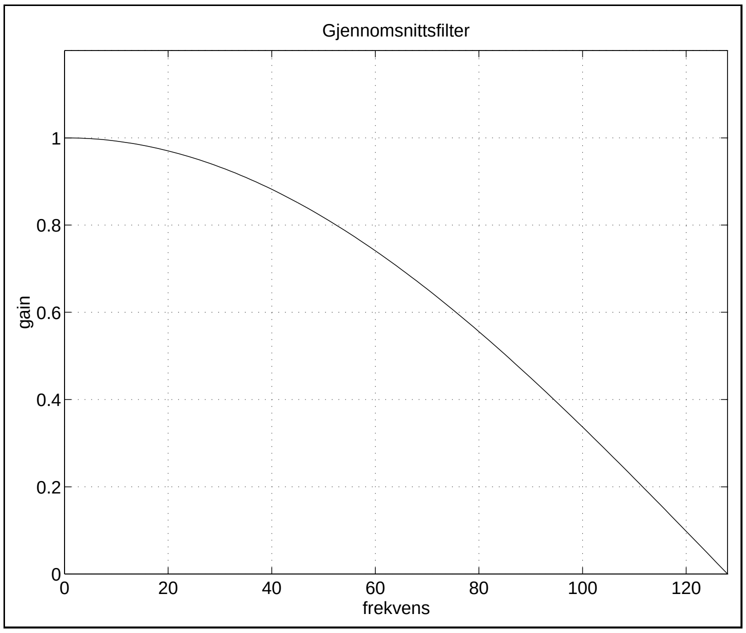
$$\begin{aligned} y_n = a_1 x_n \quad &+ \quad a_2 x_{n-1} + a_3 x_{n-2} + \cdots \\ &+ \quad b_2 y_{n-1} + b_3 y_{n-2} + \cdots \end{aligned}$$

der n er nummeret til "nåværende" sample.

La oss se på et enkelt spesialtilfelle, der vi bare bruker tidligere inngangs-samples. Vi tar simpelthen gjennomsnittet av de to forrige inngangs-samplene, og presenterer dette som et nytt utgangs-sample:

$$y_n = 0.5 \cdot x_n + 0.5 \cdot x_{n-1}$$

Siden vi tar et løpende gjennomsnitt, vil hurtige variasjoner i sampleverdiene i en viss utstrekning bli glattet ut. Dette tilsvarer at høyfrekvente komponenter blir dempet. Det ser dermed ut til at vi får et slags lavpassfilter. Sender vi således en DC-spenning gjennom filteret, hvilket tilsvarer en frekvens på 0 Hz, vil denne passere uendret gjennom filteret. Vekselvis positive og negative verdier med like stor absoluttverdi, hvilket tilsvarer en komponent på Nyquist-frekvensen, vil bli dempet fullstendig. Slik ser filterkarakteristikken ut (Nyquist-frekvensen er her 128 Hz):



Det kan overlates som en oppgave til leseren å finne ut hvorfor vi får et høypassfilter hvis vi bytter ut pluss-tegnet i likningen over med et minus-tegn.

Hvis vi i tillegg tar med tidligere utgangs-samples i differenslikningen, introduserer vi en feedback i filteret. Vi får da et *rekursivt* filter.

Det er en stor vitenskap å bestemme koeffisientene i differenslikningen slik at vi får en ønsket filterkarakteristikk i frekvensdomenet. Vi kan ikke gå nærmere inn på dette. I praksis vil man benytte oppslagsverk og tabeller der man kan finne koeffisienter for forskjellige slags filtre, der ting som senterfrekvens og båndbredde inngår som parametre.

4.3 FFT, spektralanalyse

Hvis vi har lyden representert i frekvensdomenet, kan vi lage filtre ved å behandle spekteret direkte. Vi kan gå inn og klippe bort frekvenskomponenter innenfor visse områder, og forsterke i andre områder. Det er også mange andre morsomme operasjoner vi kan utføre i frekvensdomenet.

For å overføre en samplet lyd til frekvensdomenet, må vi gjøre en *frekvensanalyse* eller *Fourier-transformasjon*. Dette er i utgangspunktet en meget tung regneoperasjon. Det var derfor en viktig oppdagelse som ble gjort av Cooley og Tukey på 1960-tallet, da de fant en svært effektiv algoritme for Fourier-transformasjon. Etterhvert har det kommet flere liknende algoritmer, som nå går under fellesbetegnelsen *Fast Fourier Transform* eller *FFT*.

FFT er en nokså innfløkt og smart algoritme, men for brukeren er resultatet greit nok: Man putter en sample-sekvens inn, og får et spektrum ut. Man kan gjerne gjøre en sammenhengende FFT på en hel lydfil, men da vil utviklingen over tid inngå i spektral-koeffisientene på en utydelig måte. Den svenske komponisten Paul Pignon har arbeidet med slik "mammut-FFT", hvor han transformerer en lydfil over i frekvensdomenet og spiller av *spekteret* som samples direkte (!). Denne sammenblandingen av tids- og frekvensdomenet er verdt å høre på, forunderlig nok.

En analysemetode som bedre følger hvordan øret og hjernen oppfatter lyd, er *korttids-FFT*. Man tar da en spektralanalyse av korte biter av lyden av gangen, slik at man får en rekke påfølgende øyeblikksbilder av spekteret som varierer over tid. Vi må da velge hvor mange *punkter* vi vil ha i FFT'en, dvs. hvor mange samples vi vil analysere av gangen. Antall punkter betegnes oftest med bokstaven N. De fleste FFT-algoritmer fungerer mest effektivt hvis N er en potens av 2. Typiske verdier som brukes er 256 (2^8), 512 (2^9), 1024 (2^{10}) og 2048 (2^{11}). Ut fra spektralanalysen vil vi da få $N/2$ verdier for amplityde og fase for de $N/2$ frekvensområdene (*frequency bins*) som er jevnt spredt fra 0 Hz til Nyquist-frekvensen. Vi ser da at vi får N tall ut til sammen, og vi puttet N tall inn. Spektralanalysen har hverken skapt eller fjernet informasjon.

Vi ser at jo flere punkter vi har i FFT'en, jo større oppløsning får vi i frekvensdomenet (vi får flere "samples" i spekteret). Med en samplingfrekvens på 32 kHz, vil $N = 512$ gi en oppløsning på 64 Hz, mens $N = 1024$ vil gi en oppløsning på 32 Hz. Men når vi øker N må vi analysere stadig lengre lydbiter, og det gjør at vi får en dårligere oppløsning i tid. $F_s = 32$ kHz og $N = 512$ vil gi 64 spektre i sekundet (hver lydbit blir 16 ms lang), mens $N = 1024$ vil gi 32 spektre i sekundet, så hver lydbit blir 31 ms lang.

Vi får dermed en avveining mellom presisjon i frekvens og presisjon i tid: Jo høyere nøyaktighet vi måler frekvensen med, jo lavere nøyaktighet må vi nødvendigvis måle tiden med. [1]

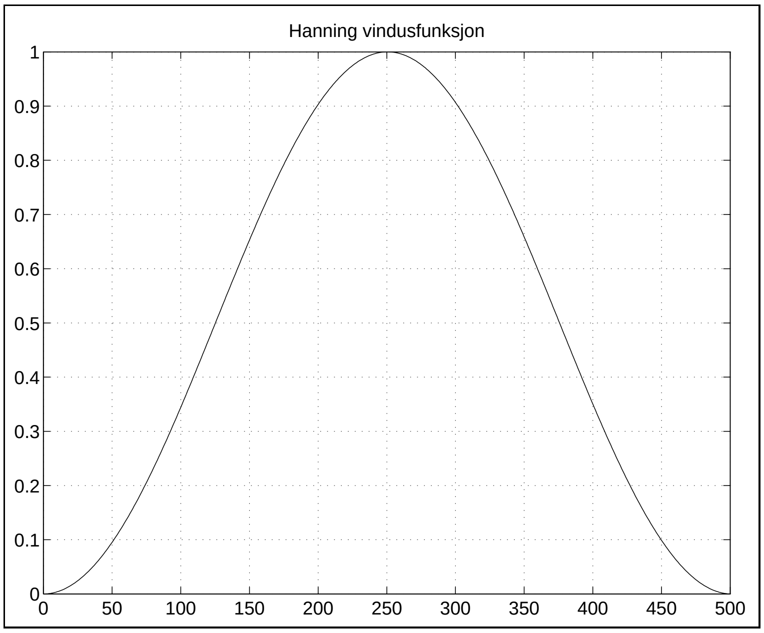
Bestemmelse av N bør gjøres i hvert enkelt tilfelle, avhengig av lyden vi arbeider med. Et stort antall tette frekvenskomponenter kombinert med lave krav til presisjon i tid (langsom dynamikk), vil typisk kreve store N. Færre frekvenskomponenter kombinert med hurtige attacks krever typisk små N. Vi ønsker generelt å sette N så liten som mulig for å få best mulig temporal oppløsning, men ikke så liten at frekvensområdene gaper over mer enn en deltone i lyden.

Vi vil imidlertid aldri *miste* informasjon i en FFT. Lyden kan derfor rekonstrueres perfekt uansett N, så lenge vi ikke har gjort noen endringer i spekteret.

4.4 Vindusfunksjoner, overlapping

De små områdene vi analyserer av gangen i en korttids-FFT kalles *vinduer*. Av tekniske grunner (undertrykking av sidebånd etc.) vil man vanligvis legge på en liten envelope på hvert av disse vinduene før man utfører FFT'en. En slik envelope kalles en *vindusfunksjon*, og det finnes mange ulike slike funksjoner, med navn som Hamming, Hanning, Kaiser, Bartlett etc. Hver av disse har sine fordeler og ulemper.

[1] I kvantemekanikken er dette kjent som *Heisenbergs usikkerhetsrelasjon*. Hvis vi betrakter partikler som bølger, vil frekvensen avhenge av partikkelens masse etter likningen $f = mc^2/h$ der f er frekvens, m er masse, c er lyshastigheten og h er Planck's konstant. Et elektron har meget liten masse og dermed meget liten frekvens. For å bestemme massen (les: frekvensen) må vi dermed betrakte elektronet over et langt tidsrom, slik vi så over. Siden elektronet beveger seg i løpet av dette tidsrommet, vil bestemmelse av posisjon og fart bli upresis. Vi kan derfor ikke angi en bestemt posisjon for elektronet, men bare plassere det innenfor et område som vi kaller en *elektronsky*. Usikkerhetsrelasjonen er gjenstand for mye mystisering og tåkelegging (New Age, Frithjof Capra etc.), men som vi ser dreier det seg om en høyst prosaisk og grei sammenheng.



For ikke å miste informasjon i overgangen mellom vinduene, er det nødvendig å la dem *overlappe*. Denne overlappingen er gjerne ganske drastisk, og derfor vil vi få ut mange flere spektre pr. sekund enn vi så på over (uten at oppløsningen i tid dermed blir noe bedre, vi får bare en slags interpolering). Datamengden bli derfor stor ut fra en korttids-FFT.

4.5 Ting man kan gjøre med et spektrum

Når vi nå har representert lyden i frekvensdomenet, åpner det seg en verden av morsomme lydbehandlingsteknikker. Vi har allerede nevnt hvor lett det er å lage filtre. Noen andre eksempler:

- Hvis vi adderer en fast verdi til hver frekvens, vil vi få et *spektralskift* som vil introdusere inharmonisitet. Dette tilsvarer på et vis en ringmodulasjon, bortsett fra at vi får ett sidebånd, ikke to.
- Hvis vi multipliserer hver frekvens med en fast verdi, får vi et pitch-skift uten medfølgende hastighetsendring. Dette er en metode for harmonizing som gir meget god kvalitet.
- Vi kan snu spekteret opp-ned, slik at grunntonen blir den høyeste harmoniske i spekteret og omvendt, eller permutere deltonene på andre måter.
- Vi kan sortere deltonene etter amplityde, og beholde bare de kraftigste.
- Vi kan filtrere *spekteret*, enten langs frekvensaksen (vil framheve eller dempe bratte flanker i hvert korttidsspektrum) eller langs tidsaksen (vil framheve eller dempe endringer i klangen over tid).

- Vi kan analysere *to* lyder, og f.eks. multiplisere spektrene med hverandre eller lage glidende overganger mellom dem på forskjellige måter ("morfing").
- Vi kan tvinge deltonene henimot visse frekvenser, f.eks. en musikalsk skala.

Her er det fritt fram for fantasien! Parametrene til disse metodene kan selvsagt variere over tid.

4.6 IFFT, resyntese

Etter at vi har vrengt spekteret på en eller annen fantastisk måte, er vi nødt til å komme oss tilbake til tidsdomenet før vi kan høre lyden. Dette kalles *resyntese*, og er egentlig bare en additiv syntese. Denne resyntesen kan gjøres med den samme algoritmen som FFT, men med et par små endringer som gjør at den virker *baklengs*. Dette heter *invers* FFT (IFFT).

Nokså ofte vil vi ha gjort endringer i spekteret som gjør at vinduene ikke "butter sammen" helt som de skal i resyntesen. Dette hører man lett som knitring og "firkantet" lyd. Det er da nødvendig å interpolere mellom analyseverdiene for hvert vindu, og vi må ty til resyntese ved rå makt, dvs. addere et stort antall sinustoner direkte. Dette er langt mer tidkrevende enn IFFT.

Hvis vi strekker oppdateringen av parametrene i resyntesen utover i tid, får vi en time-stretch (kompresjon/dilatasjon) av høy kvalitet uten medfølgende pitch-endring.

4.7 Fase-vokoderen

Det ble nevnt at tallene som kommer ut av en FFT er amplityde og fase for den deltonen som ligger innenfor hvert frekvensområde. [2] Men musikere er nok mer opptatt av *frekvens* enn av fase. Derfor benyttes ofte en teknikk som heter *fase-vokoding*, der faseinformasjonen fra FFT'en brukes til å beregne frekvens. Dette baserer seg på at endringen i fase fra analysevindu til analysevindu er proporsjonal med deltonens frekvens. Vi får da ut eksakte frekvenser, ikke bare informasjon om at deltonen ligger innenfor en bestemt bin.

Musikkprogrammet C-Sound har en fasevokoder, og den finnes også i SoundHack for Mac og flere andre systemer. Fasevokoderen er brukt i mange musikkstykker til mange forskjellige formål.

4.8 Lineær-prediktiv koding

Menneskets tale fungerer ved at stemmebåndene produserer et pulstog med et nokså flatt spektrum. Dette pulstoget filtreres videre av hals og munnhule. Forskjellige vokaler tilsvarer forskjellige frekvenskarakteristikker for dette filteret. Et typisk trekk ved disse karakteristikkene er to eller flere tydelige resonanser (formanter).

[2] Vi håper at det er bare en slik sinusoid komponent i hver bin, men dette vil ofte ikke slå til (f.eks. ved hvit støy)

Ved lineær prediksjon forsøker vi å produsere et filter av differenslikning-typen, med koeffisienter som varierer over tid. Dette filteret skal etterlikne filterkarakteristikken til lyden vi analyserer. Signalet kan siden re-syntetiseres ved å eksitere filteret med et syntetisk pulstog som etterlikner stemmebåndene. Vi kan på den måten kode tale bare gjennom filterkoeffisientene, som varierer forholdsvis langsomt, og dermed har vi en effektiv metode for data-reduksjon av talesignaler. Lineær-prediktiv koding (LPC) har oppnådd en posisjon innen talesyntese, selv om lydkvaliteten ofte er dårlig (dette bunner i bruk av for få filterkoeffisienter, og stiliseringen av eksitator). Den typiske "robot-stemmen" som var vanlig i tale-synthesizere for noen år siden, var produsert ved hjelp av LPC.

Som vi har sett, kan eksitator-resonator-modellen anvendes på de fleste musikkinstrumenter. LPC begrenser seg derfor ikke til bruk på tale.

LPC kan brukes til å vri lyden på mange rare måter. Vi kan f.eks. variere pitch ved å skru på frekvensen til eksitator. I vanlige harmonizere, og i pitch-skift med FFT, vil formantenes posisjon endre seg. Tale fungerer ikke på den måten, og LPC gir derfor et mer korrekt resultat, uten den typiske "Donald-stemmen".

Vi kan også kjøre andre lyder gjennom filteret vi har generert. På den måten kan vi gjøre kryss-syntese, f.eks. ved at vi simulerer at vi har et symfoniorkester plassert i strupehodet istedenfor stemmebånd. Vi får da et "talende orkester".

Det må innrømmes at LPC ofte ikke låter så bra. Det krever erfaring å utnytte LPC på den beste måten, bestemme hvor mange filterkoeffisienter vi skal ta med etc.

Guruen innenfor musikalske anvendelser av LPC er den amerikanske komponisten Paul Lansky. Musikkprogrammet C-Sound inneholder en implementasjon av LPC.

4.9 Filterbanker

Det finnes en annen metode for spektralanalyse som kanskje er lettere å begripe, men som ikke er så beregningseffektiv som FFT, nemlig bruk av en *filterbank*. En filterbank er en bunke med båndpassfiltre koblet i parallell, med senterfrekvenser som er spredt over hele frekvensområdet. Det gjøres en RMS-analyse eller annen type energi-analyse på output fra hvert filter, og dermed får vi informasjon om hvor mye energi vi har innenfor hvert frekvensbånd.

Filterbanker er historisk viktige, fordi de kan implementeres analogt. De fleste enkle spektralanalysatorer av den typen vi finner i lydstudioer, fancy stereoanlegg etc. benytter filterbanker. Den klassiske effektboksen som kalles "vokoder" har en filterbank koblet sammen med en oscillatorbank. Dette gir en enkel form for analyse-resyntese.

I de senere år har filterbanker fått en kraftig renessanse. Man har begynt å sette pris på *kontinuiteten* som filterbanker gir i forhold til den mer opphakkede, vindusbaserte FFT-metodikken. Samtidig har digitale signalprosessorer blitt raske nok til å benytte filterbanker i sann tid. De nye digitale kompakt-kassettene (DCC) benytter filterbanker som en del av datakompresjonen.

Kapittel 5

Granulære metoder

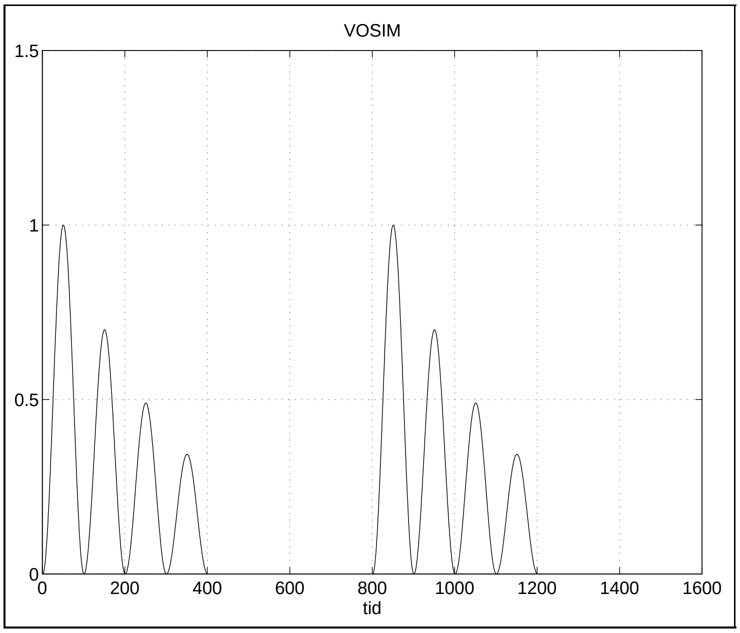
I dette kapittelet skal vi se på en måte å betrakte lyd på som kan virke uvant etter alt snakket om kontinuerlige sinusoider og varierende spektre. Det går an å se på lyd som et aggregat av utallige små "korn" (grains) som er plassert rundt i frekvens og/eller tid. Vi kan syntetisere lyd ved å pøse på med slike korn, vi kan forsøke å finne ut hvilke korn av en valgt type vi må benytte for å bygge opp en lyd vi analyserer, eller vi kan klippe opp en lyd i små biter som vi behandler på ulike måter. Alt dette kan vi kalle *granulære* lydbehandlingsmetoder, etter det latinske ordet *granula* (korn). [1] Hva slags korn vi vil benytte i f.eks. granulær syntese er igrunnen valgfritt. Akustisk vil det gjerne dreie seg om små "plopp" av type dryppende vannkran eller vibrafon. Disse ploppene kan f.eks. spres ved hjelp av tilfeldighetsgeneratorer i "skyer". Å bygge opp en lyd med slike korn har en parallell i *pointilismen*. Dette var en retning innen impresjonismen der man malte bilder ved å avsette tusener av små fargepunkter på lerretet.

5.1 VOSIM-syntese

VOSIM-syntese (for *VOice SIMulation*) er en granulær syntesemetode som opprinnelig ble laget for talesyntese. Kornene har her en spesiell form med parametre som vi kan velge. Hvert korn består av et lite pulstog med N klokkeformede pulser (kvadrert sinus er fint), hver med en pulsbredde w. Etter den første pulsen multipliseres påfølgende pulser med en dempningsfaktor $d < 1$. Hele pulstoget gis en amplityde a, og så gjentas kornene med en periode L som bestemmer grunntonefrekvensen ($f = 1/L$).

Her er en bit av et VOSIM-signal, med to korn:

[1] Bergarten *granitt* består av bittesmå korn, og har fått sitt navn av den samme stammen



Ved hjelp av parametrene kan vi styre pitch, amplityde og klangfarge. En enkelt VOSIM-generator lager et spektrum med en enkelt formant. For flere formanter, bruker vi flere VOSIM-generatorer.

VOSIM ble utviklet ved Institutt for Sonologi i Utrecht, Nederland. Metoden har ikke noen stor utbredelse i dag.

5.2 FOF-syntese

En *FOF* (Forme Onde Formantique), er et lite korn som består av en sinus med en eksponensiell envelope. På samme måte som i VOSIM, vil syntese med en enkelt FOF-generator gi et spektrum med en enkelt formant. Miksing av flere ulike FOF-generatorer gir et spektrum med flere formanter, som kan brukes til syntese av f.eks. tale og sang.

En FOF kan tolkes som en enkelt impuls fra stemmebåndene, som er filtrert av hals og munnhule. Hvis man trekker pusten inn, kan man klare å lage enkeltstående FOF'er som kan høres separat [2]. En slik liten kurveform, som er filterets respons på en impuls, kalles (gjett!) filterets *impulsrespons*. Et filter er fullstendig karakterisert ved sin impulsrespons; ja faktisk er frekvenskarakteristikken ikke noe annet enn Fouriertransformasjonen til impulsresponsen. Ved invers Fourier-transformasjon kan vi dermed gå fra frekvensrespons til impulsrespons. Hvis vi sampler denne impulsresponsen, kan vi benytte samplene direkte som filterkoeffisienter i et ikke-rekursivt filter. (Dette avsnittet var en digresjon, men det ville være en skandale om ordet "impulsrespons" ikke ble nevnt i en tekst om digital lydbehandling).

Programmet *Chant* fra IRCAM benytter FOF-syntese. Det finnes en meget imponerende syntese av "Nattens Dronning" av Mozart som er laget med dette systemet. C-Sound inneholder også en FOF-generator.

[2] Det er den engelske komponisten Trevor Wishart som har pekt på dette

I forbindelse med granulær syntese må vi også nevne amerikaneren Barry Truax og svensken Peter Lundén, som begge har laget stykker som illustrerer metoden godt. "Riverrun" av Truax anbefales spesielt.

5.3 Granulering med vindusfunksjoner

Vi kan "granulere" en lyd ved å klippe den opp i små biter med en varighet på typisk mellom 20 og 500 millisekunder. Disse bitene kan så komprimeres eller strekkes, forsterkes eller dempes, filtreres etc. uavhengig av hverandre, eller de kan omkastes (permuteres) på forskjellige måter. Det er ikke lurt å klippe av lydkornene brått. Hvis vi f.eks. bytter om rekkefølgen kan vi da risikere at kurven gjør sprang i overgangen mellom kornene. Slike diskontinuiteter gir klikk. Vi legger derfor på envelopes på hvert lille korn, som faller pent ned mot null i hver ende. Det benyttes samme type vindusfunksjoner som vi så på under diskusjonen om FFT (Hanning etc.). Det er vanlig å la vinduene overlappe.

Ved å strekke og komprimere kornene kan vi gjøre pitch-skift og time-stretch med relativt lav kvalitet. Nesten alle kommersielle harmonizere benytter granulering med vindusfunksjoner. Det begynner nå å komme effektbokser som utfører morsomme granuleringsmetoder med tilfeldighet etc. IRCAM-arbeidsstasjonen er også velegnet til granulering, og dette har vært brukt av bl.a. Philippe Manoury.

5.4 Granulering med nullgjennomgangsanalyse

Hvis vi passer på å klippe opp lyden akkurat på de tidspunktene der kurven passerer nullnivået på vei oppover, kan vi unngå klikk uten å modulere med vindusfunksjoner. Dette medfører at kornene vil variere i lengde: I perioder med høye frekvenser eller støylyder får vi mange nullgjennomganger og små korn, mens i perioder med lave frekvenser, stillhet eller fattig spektrum får vi store korn.

En spesiell effekt får vi ved å avspille disse kornene i et spiralformet mønster på denne måten:

1 2 3 4 5, 3 4 5 6 7, 5 6 7 8 9, 7 8 9 10 11, etc.

Her er *inkrementet* satt til 2 nullgjennomganger, mens *vindus-størrelsen* er satt til 5 biter mellom påfølgende nullgjennomganger. Dette vil strekke lyden ut i tid, og gi artige effekter.

Det var den sveitsiske komponisten Rainer Boesch som gjorde oss i Norge oppmerksom på denne typen teknikker, og vi har siden benyttet dette flittig. I stykket "Strøk" av Rolf Wallin er hardingfelemusikk behandlet ved hjelp av granulering med nullgjennomgangsanalyse. Den norske lydbehandlingsmaskinen "Fredag" ble i sin tid konstruert med spesiell hardware og software som kunne (og forsåvidt fortsatt kan) utføre dette i sann tid med endel spesielle fiduser.

5.5 Granulering og FFT

Ved korttids-FFT betrakter man små biter av lyden av gangen. Hver av disse bitene blir dekomponert i sine enkelte sinusoide bestanddeler. Analysen gir oss derfor et stort antall små sinusoide biter som er plassert i tid og frekvens. Vi kan dermed betrakte korttids-FFT som en metode som finner ut hvilke sinusoide korn vi må bruke for å bygge opp lyden. Invers FFT kan da betraktes som en granulær syntese som resyntetiserer lyden ut fra informasjonen om alle disse små kornene.

Dette er en litt uvanlig måte å se på korttids-FFT på. Når vi tenker i granulære baner, blir det naturlig å gjøre ting som å plukke vekk korn, permutere korn i tid og frekvens etc. Vi kan også resyntetisere med andre typer korn enn sinusoider, f.eks små plinger generert av Karplus-strong-algoritmen.

NoTAM bedriver utvikling av et lite program for våre SGI-maskiner som retter seg inn mot å betrakte korttids-FFT som en granulær metode. Arbeidstittelen er *Ceres*, etter den romerske korn-gudinnen.

5.6 Wavelets

Det går an å forsøke å dekomponere en lyd i andre typer korn enn små sinusoider. En matematiker vil si at vi benytter en annen *ortogonal basis* for dekomposisjonen. I signalbehandling og matematikk benyttes betegnelsen *wavelets* om våre "korn" ("wavelet" er diminutiv av "wave", og betyr altså "liten bølge"). Det finnes flere ulike klasser av wavelets i bruk. Det finnes også forskjellige algoritmer for å dekomponere en funksjon opp i enkelte wavelets, oftest er disse basert på en matematisk operasjon som heter *konvolusjon*.

Wavelets har vært et hett tema i matematikk og signalbehandling i de senere år. Det viser seg at denne betraktningsmåten har mye for seg både teoretisk og i anvendelser. Støyreduksjon ved wavelet- transformasjon er nå en viktig teknikk.

Syntese og analyse med generelle wavelets begynner også å få musikalske anvendelser.

Kapittel 6

Spatialisering

For å gi "liv" til en lyd ønsker vi gjerne å simulere at lydkilden er plassert et visst sted i et rom med en viss romklang. Dette heter *spatialisering*.

6.1 Ekko

Ekko simuleres ved hjelp av en *digital delay* som forsinker lyden. Den forsinkede lyden (ekkoet) mikses så med originalen. Dersom vi setter forsinkelsestiden til en liten verdi, vil vi få vekselvis konstruktiv (signalene er i fase) og destruktiv (signalene er i motfase) interferens mellom det opprinnelige og det forsinkede signalet på forskjellige frekvenser. Dette betyr at oppsettet fungerer som et *kamfilter*, ja mer enn det, det *er* ikke noe annet enn et kamfilter, med differenslikningen

$$y_n = x_n + x_{n-m}$$

der m er antall samples forsinkelse. Avstanden mellom toppene i filterkarakteristikken er gitt ved lengden på forsinkelsen.

Vi kan så multiplisere utgangssignalet med en dempningsfaktor $d < 1$, og sende dette tilbake i inngangen (feedback). Vi får da en effekt av typen Hei Hei Hei Hei Hei Hei. Hvis forsinkelsen er i størrelsesorden 50 ms, vil dette gi en metallisk og kunstig romklang. Dette er fremdeles et filter, nå av den rekursive typen, med differenslikning

$$y_n = x_n + x_{n-m} + d \cdot y_{n-m}.$$

6.2 Romklang

Ekkomaskinen med feedback fra forrige avsnitt kan brukes til å simulere romklang i et stort metallrør eller noe slikt. De fleste naturlige rom har en mye mer komplisert geometri, med forskjellige typer materialer som reflekterer lyden på ulik måte. Dette gjør at romklangen får et meget komplisert forløp.

Vi kan måle impulsresponsen til en konsertsal ved å fyre av en pistol, og skrive ut ekkoene som pulser som funksjon av tiden. Vi kan så bruke denne impulsresponsen til å konstruere et digitalt filter direkte. Vi sier da at vi *folder* eller *konvolverer* lyden med impulsresponsen. Hittil har slik "ekte" romklang vært altfor beregningskrevende for praktisk bruk. En etterklangstid på 10 sekunder vil f.eks. kreve en million addisjoner og multiplikasjoner for hvert eneste sample i lyden (ved 50 kHz samplingfrekvens og stereo). Ny teknologi som bl.a. er beregnet for Virtual Reality-anvendelser (f.eks. "Convolvotron") begynner imidlertid å gjøre slike ting mulig.

For å forenkle beregningene, stiliserer vi impulsresponsen endel. Som oftest vil impulsresponsen begynne med endel enkeltstående ekko ("early response") som tilsvarer de primære og sekundære refleksjonene fra vegger og tak. Disse ekkoene kan vi simulere med digitale delays. Etterhvert som klangen dør ut, blir impulsresponsen meget kompleks. Vi får da inn alle mulige kombinasjoner av tidligere ekko som reflekteres enda en gang. Det er vanlig å benytte fiffig oppsatte allpassfiltre for å simulere dette.

Hver gang lyden reflekteres eller passerer gjennom noen meter luft vil den miste diskant. Digitale delays med feedback og et lavpassfilter i tilbakekoblingen vil kunne simulere dette. [1]

6.3 Plassering av lydkilder i rommet

I heftet om akustikk så vi på hvordan øret og hjernen bestemmer retningen og avstanden til en lydkilde. For å simulere *avstand* bruker vi disse effektene:

- Endring av amplityde, ved at amplityden faller lineært med avstanden (effekten, som avhenger av kvadratet av amplityden, vil da falle med kvadratet av avstanden).
- Endring av frekvensinnhold, ved at lyden mister mer og mer diskant ettersom avstanden øker (simuleres med et lavpassfilter)
- Doppler-effekten. Dette simuleres ved å sette opp en digital delay som simulerer tiden lyden bruker fra lydkilden til lytteren. Ettersom avstanden endres, vil da også delaytiden endres, og Dopplerske pitch-skift oppstår da helt "av seg selv".

For å simulerer *retning* bruker vi disse effektene:

- Amplityde-panorering mellom kanalene
- Faseforskjell og Haas-effekt (simuleres med digitale delays)
- Eventuelt simulering av ytterørets og hodets filtrerende effekt avhengig av retning

[1] Hvis delay-tiden settes meget liten, er vi tilbake i Karplus- Strong-algoritmen!

6.4 Andre effekter

Chorus kan simuleres ved å mikse et antall parallelle systemer der lyden gjennomgår en langsomt varierende, tilfeldig forsinkelse. Dette vil simulere både at de separate lydkildene er noe ute av fase, og at de har noe ulik pitch (på grunn av de resulterende Doppler-skiftene). I tillegg kan vi legge på en tilsvarende tilfeldig modulasjon av amplityde.

Flanger er ikke noe annet enn en langsom endring av delay-tid, gjerne modulert av en lavfrekvent sinus.

Fuzz lages ved å kjøre lyden gjennom en ulineær overføringsfunksjon (se avsnittet om ulineær forvrengning).