CSOUND

Øyvind Hammer
Norsk nettverk for Teknologi, Akustikk og Musikk

Oslo, 25. august 1994

Innhold

1 Innledning **2**

2 Et lite eksempel **3**
- 2.1 Header . . . 3
- 2.2 Instrumentet . . . 4
- 2.3 Partituret . . . 4

3 Flere triks **5**
- 3.1 P-felter . . . 5
- 3.2 Envelopes, GENs, carry . . . 6

4 Synteseteknikker **8**
- 4.1 Ringmodulering . . . 8
- 4.2 FM-syntese . . . 9
- 4.3 Subtraktiv syntese . . . 9
- 4.4 Waveshaping . . . 10
- 4.5 FOF . . . 11
- 4.6 Karplus-Strong . . . 12

5 Lydbehandling **13**
- 5.1 Kor . . . 13
- 5.2 Global romklang . . . 14
- 5.3 Fasevokoding . . . 15
- 5.4 Ceres . . . 15
- 5.5 Lineær-prediktiv koding . . . 15
- 5.6 Additiv deltoneanalyse/resyntese . . . 16
- 5.7 Videre . . . 17

Kapittel 1

Innledning

Heftet gir en kortfattet innføring i Csound, som er et "klassisk" program for syntese og omforming av lyd. Csound ble opprinnelig skrevet av Barry Vercoe ved MIT, og føyer seg inn i rekken av beslektede systemer helt fra Max Mathews' *Music 4* på 1960-tallet, via *Music 4B*, *Music 4C*, *Music V*, *Music 11* og *Cmusic*. Csound brukes av svært mange komponister i dag, og er tilgjengelig gratis for UNIX-maskiner, Macintosh og IBM PC. En gjennomgang av Csound er obligatorisk i nesten alle kurs i computermusikk rundt om i verden.

Csound er ikke et generelt programmeringsspråk, men et språk for å definere oppbyggingen av instrumenter fra primitive byggeblokker. På mange måter kan Csound sammenliknes med *Max*, hvor man også kobler sammen byggeklosser til større enheter. Csound har imidlertid i utgangspunktet ikke noe grafisk brukergrensesnitt, så instrumentene må bygges opp tekstlig med en syntaks som nok dessverre virker litt kryptisk i begynnelsen.

Med Csound er det nesten ingen grenser for hvor komplekse ting man kan gjøre. Hvis man ønsker et FM-instrument med tusen operatorer, er det helt i orden. Vanligvis kjøres ikke Csound i sann tid, så arbeidsgangen er at man først lager den tekstlige beskrivelsen. Deretter starter man Csound, som sender den produserte lyden til en lydfil for senere avspilling.

Du må gi *to* tekstfiler som input til Csound. I den ene filen, *orkesterfilen*, ligger en beskrivelse av de forskjellige instrumentene, hvor du har angitt sammenkoblingen av byggeklosser som oscillatorer, filtre etc. I den andre filen, *partiturfilen* eller (*score file*), ligger en liste med noter, dvs. oppkall av instrumentene til ulike tider og med ulike parametre. Partiturfilen kan godt sammenliknes med en MIDI-fil, og det finnes programmer som kan konvertere mellom MIDI-filer og partiturfiler. I partiturfilen angis også de forskjellige *tabellene* som Csound skal bruke til kurveformer, enveloper etc. Hvis du kan programmere litt i C, kan du lage et C-program som skriver partiturfiler etter en eller annen algoritme som du finner på (algoritmisk komposisjon).

For å få noe særlig utbytte av Csound, må du ha en god forståelse av prinsippene for lydsyntese og lydbehandling. Du må vite hva FM-syntese, additiv syntese, filtrering etc. er, og du må vite ting som f.eks. hvilke matematiske operasjoner som benyttes for å mikse lyder eller legge en envelope på en lyd. Dette er gjennomgått i tidligere teorikurs på NoTAM. Kompendiene fra disse kursene er tilgjengelige.

Kapittel 2

Et lite eksempel

For å lage en liten lyd med Csound, må vi først definere et instrument. Dette gjøres i orkesterfilen, som vi kan kalle `eksempel.orc`. Start opp en teksteditor med kommandoen `jot eksempel.orc` på Indy. På Mac skulle *Word* eller noe slikt være brukbart.

Skriv inn følgende (forklaring under):

```
sr     =  16000
kr     =  1000
ksmps  =  16
nchnls =  1

instr 1
  a1 oscil 10000,440,1
out a1
endin
```

2.1 Header

Først i orkesterfilen må vi ha en *header*, der vi spesifiserer samplingfrekvens og et par andre rare ting. `sr = 16000` betyr at vi ønsker en samplingfrekvens på 16 kHz. Dette er ikke mye, men nok til dette enkle eksemplet (beregningene går fortere jo lavere samplingfrekvens vi bruker). Ulike maskiner har litt ulike sett av tillatte samplingfrekvenser, på Indy kan man velge mellom 8000, 11025, 16000, 22050, 32000, 44100 og 48000.

Et viktig poeng i Csound er skillet mellom *samplingrate* og *kontrollrate*. Samplingraten er antall samples i lyden pr. sekund, og vi kunne godt ha beregnet alle tallene i instrumentet med denne hastigheten. Ofte har man imidlertid parametre som endrer seg noe langsommere, f.eks. enveloper. For å minske beregningstiden, kan man bestemme at disse verdiene skal oppdateres med en lavere hastighet. Denne lavere "samplingfrekvensen" kalles kontrollrate, og vi setter den her til 1000 Hz med kommandoen `kr = 1000`. Vi må også angi forholdet mellom samplingraten og kontrollraten med kommandoen `ksmps = 16`. Dette er selvsagt idiotisk, maskinen burde kunne klare denne divisjonen selv.

Kommandoen `nchnls = 1` angir at vi skal lage en mono-lydfil.

2.2 Instrumentet

Man kan ha mange forskjellige instrumenter i et orkester, og disse nummereres i orkesterfilen. Dette gjøres i innledningen til instrumentdefinisjonen, her med kommandoen `instr 1`. Instrumentet avsluttes med kommandoen `endin`.

"Boksene" som vi skal koble sammen i et instrument angis på separate linjer. Hver linje begynner med et *variabelnavn* som brukes som referanse til output fra boksen. Så følger boksens *type*, med eventuelle typespesifikke *parametre*. De forskjellige boks-typene og deres parametre er beskrevet i Csound-manualen.

`a1 oscil 10000,440,1` betyr således at vi setter opp en boks av type *oscil*, som er en generell oscillator. Den tar tre parametre, nemlig amplityde, frekvens og kurveform. Amplityden angis som et heltall, der 32767 er fullt utslag. Frekvensen oppgis i Hz. Kurveformen oppgis med et tabellnummer som refererer til en tabell som vi senere skal definere i partiturfilen.

Output fra oscillatoren sendes til variabelen `a1`. Ved å la variabelnavnet begynne med en *a*, angir vi at variabelen skal oppdateres for hvert sample (*"audio rate"*). Vi kunne dermed godt ha kalt variabelen "asparges", men ikke "potet". Hvis navnet hadde begynt med *k*, ville den blitt oppdatert med kontrollhastighet, dvs. her 1000 ganger i sekundet.

`out a1` betyr at output fra instrumentet skal hentes fra variabelen `a1`.

Vi er nå ferdig med orkesterfilen.

2.3 Partituret

Så må vi lage partiturfilen. Vi kan kalle den `eksempel.sco`. Skriv inn følgende:

```
f1  0  4096  10  1
i1  0  4
e
```

`f1` betyr at vi skal definere funksjonstabell nummer 1, som vi refererte til i orkesterfilen. Tabellen skal opprettes ved tidspunkt 0 sek, og den skal inneholde 4096 samples (må være en potens av 2). Vi skal bruke en subrutine som heter `GEN10`, derfor tallet 10. `GEN10` tar som parametre de relative amplitydene til en overtonerekke; her spesifiserer vi bare en grunntone, slik at tabellen fylles med en sinusperiode.

Så følger alle "notene", dvs. oppkall av instrumenter til forskjellige tidspunkter. Her har vi bare en note; `i1` betyr at vi skal bruke instrument nummer 1. Noten skal starte ved tidspunkt 0 sek, og skal vare i 4 sekunder.

`e` angir slutten på partiturfilen.

Vi kan nå starte Csound, på Indy gjøres det med kommandoen `csound -Ao eksempel eksempel.*`. Dette er litt mer inngående forklart i heftet *Lokal guide til lydbehandling i UNIX*. Csound skal nå produsere en lydfil med navnet *eksempel*. Spill lyden med f.eks. `sfplay /sounds/eksempel`.

Som man ser, er syntaksen i Csound meget kortfattet, konsis og kryptisk. Det tar tid å venne seg til dette, men fordelen er at det blir mindre å skrive.

Kapittel 3

Flere triks

3.1 P-felter

Instrumentet i forrige kapittel kunne bare spille en fast frekvens med en fast lydstyrke. Det må vi fikse. Parametre i instrumentet kan styres fra verdier som vi oppgir i partiturfilen. Parametrene til `i`-kommandoene i partiturfilen heter `p1`, `p2` osv. De tre første *p-feltene* har en fast betydning:

$$\begin{aligned} p1 &= instrumentnummer \\ p2 &= starttidspunkt \\ p3 &= varighet \end{aligned}$$

(se eksemplet i forrige avsnitt).

De resterende p-feltene (p4, p5 etc.) kan vi benytte som vi vil. Vi kan la p4 styre frekvens og p5 amplitude. Det finnes en funksjon som heter `cpspch` som gjør om fra formatet *oktav.pitchklasse* til en frekvens i Hz. Funksjonen `ampdb` gjør om fra en dB-verdi til en amplitude.

Orkesterfilen kan da skrives slik:

```
sr     =  16000
kr     =  1000
ksmps  =  16
nchnls =  1

instr 1
  a1 oscil ampdb(p5),cpspch(p4),1
out a1
endin
```

Og partiturfilen kan se slik ut (legg merke til at semikolon markerer en kommentar):

```
f1  0  4096  10  1
;   p2 (start)    p3 (varighet)    p4 (frekvens)    p5 (styrke)
i1  0             0.1              8.00             50
i1  0.2           0.1              8.00             55
i1  0.4           0.1              8.00             60
i1  0.8           0.1              8.04             65
i1  1.2           0.1              8.07             70
i1  1.6           1                9.00             70
i1  1.6           1                9.04             70
i1  1.6           1                9.07             70
e
```

3.2 Envelopes, GENs, carry

Vi fortsetter å bygge ut instrumentet fra forrige avsnitt. For det første vil vi gjerne legge en *omhyllingskurve* (envelope) på hver tone. Vi blar litt i Csound-manualen, og finner "boksen" (enhetsgeneratoren) `linen`. Formatet er definert slik:

```
kr linen kamp,irise,idur,idec
ar linen xamp,irise,idur,idec
```

Typisk Csound-kryptisk! Hva betyr dette? Det man forsøker å få fram er at `linen` kan produsere sin output enten med "kontrollrate" eller med "audio rate" (samplingfrekvensen). Attack-tiden ("rise") kan oppdateres med "initialiseringsrate", dvs. hver gang instrumentet kalles opp fra en `i`-kommando i partituret. Det sammet gjelder "dur" og "dec" (release-tiden). Amplityden kan oppdateres med kontrollrate eller langsommere dersom output oppdateres med kontrollrate. Hvis output oppdateres med audio rate, kan amplityden oppdateres med alle slags hastigheter (angitt med "x").

Vi velger k-versjonen, og sender output fra `linen` inn i amplityde-parameteren til oscillatoren. I tillegg lar vi kurveform-parameteren styres fra et nytt p-felt, slik at vi kan endre klangfargen for hver note:

```
sr     =  16000
kr     =  4000
ksmps  =  4
nchnls =  1

instr 1
  k1    linen   ampdb(p5),0.05,p3,p3*0.4
  a1    oscil   k1,cpspch(p4),p6
        out     a1
endin
```

I partiturfilen definerer vi en funksjonstabell nummer 2 ved hjelp av `GEN10`, med et rikere spektrum (8 overtoner), og legger til et nytt p-felt p6. I tillegg utnytter vi en nyttig finesse i syntaksen, nemlig *carry-tegnet* punktum. Et punktum betyr at vi skal bruke samme verdi som i forrige linje. De siste punktum på en linje kan utelates. Det finnes flere andre carry-tegn, bl.a. *pluss*, som betyr at vi skal bruke p2+p3 fra forrige linje, dvs. tidspunktet for forrige notes slutt (brukes ikke her).

```
f1  0  4096  10  1
f2  0  4096  10  1 .6 .3 .1 .3 .6 .3 .1
;   p2 (start)  p3 (varighet)  p4 (frekvens)  p5 (styrke)  p6 (kurve)
i1  0           0.1            8.00           50           1
i1  0.2         .              .              55
i1  0.4         .              .              60
i1  0.8         .              8.04           65
i1  1.2         .              8.07           70
i1  1.6         1              9.00           .            2
i1  .           .              9.04
i1  .           .              9.07
e
```

Kapittel 4

Synteseteknikker

Dette kapittelet presenterer endel enkle Csound-instrumenter som demonstrerer ulike syntesemetoder.

4.1 Ringmodulering

I dette eksemplet ser vi hvordan man kan gjøre *ringmodulering*, dvs. at vi multipliserer to lyder med hverandre. Vi velger å modulere en lyd med et rikt spektrum (carrier) med en sinustone hvor frekvensen varierer (modulator):

```
sr     =  32000
kr     =  2000
ksmps  =  16
nchnls =  1

instr 1
  kmodf linen   40,p3*0.4,p3,p3*0.4       ; Styrer frekvensen til modulator
  kcara linen   10000,p3*0.1,p3,p3*0.1   ; Styrer amplityden til carrier
  amod  oscil   10000,kmodf,1             ; Modulator
  acar  oscil   kmoda,100,2               ; Carrier
        out     amod*acar/10000
endin
```

(Legg merke til at vi må skalere output ved å dividere med 10000, for å unngå klipping).

```
f1 0 4096 10 1
f2 0 4096 10 1 0.6 0.3 0.1 0.1 0.3 0.6 0.1
i1 0 20
e
```

4.2 FM-syntese

FM er lett å få til i Csound, f.eks. slik:

```
sr     =  32000
kr     =  2000
ksmps  =  16
nchnls =  1

instr 1
  kmodf linen   400,p3*.05,p3,p3*.95     ; Styrer amplityden til modulator
  kcara linen   10000,p3*.05,p3,p3*.95   ; Styrer amplityden til carrier
  amod  oscili  kmodf,100,1
  acarr oscili  kcara,100+amod,2
        out     acarr
endin
```

Vi bruker her `oscili` istedenfor `oscil`, fordi dette minsker interpolasjonsstøyen.

Partiturfilen er den samme som i forrige eksempel.

4.3 Subtraktiv syntese

Subtraktiv syntese betyr filtrering av en lyd med et rikt spektrum. I orkesterfilen under bruker vi `reson`, som er et båndpassfilter med styrbar senterfrekvens og båndbredde.

For å holde lydstyrken på det opprinnelige nivået selv etter filtreringen, bruker vi `balance`, som her sørger for at `afilt` holdes på samme nivå som `abuzz`.

```
sr     =  16000
kr     =  2000
ksmps  =  8
nchnls =  1

instr 1
  kbuzz linen   10000,p3*.1,p3,p3*.1         ; Styrer amplityden
  kfilt linseg  2000,p3*0.5,0,p3*0.5,2000    ; Styrer senterfrekvens
  abuzz oscili  kbuzz,100,1                  ; Tonegenerator
  afilt reson   abuzz,kfilt,100              ; Filter
  abal  balance afilt,abuzz                  ; Automatisk gain-kontroll
        out     abal
endin
```

For å få et rikt spektrum på den opprinnelige lyden, bruker vi `GEN11` i partiturfilen. `GEN11` gir et flatt spektrum, og vi ber om å få med 50 harmoniske:

```
f1 0 8192 11 50
i1 0 20
e
```

Lydkilden kan byttes ut med en hvitstøygenerator, slik:

```
sr     =  16000
kr     =  2000
ksmps  =  8
nchnls =  1

instr 1
  knois linen   10000,p3*.1,p3,p3*.1         ; Styrer amplityden
  kfilt linseg  2000,p3*0.5,0,p3*0.5,2000    ; Styrer senterfrekvens
  anois rand    knois                        ; Tonegenerator
  afilt reson   anois,kfilt,100              ; Filter
  abal  balance afilt,anois                  ; Automatisk gain-kontroll
        out     abal
endin
```

4.4 Waveshaping

Følgende er basert på et instrument laget av Rob Waring. `expseg` lager en stykkevis eksponensiell funksjon. Parametrene til `expseg` er startverdi, tid til verdi 2, verdi 2, tid til verdi 3, etc.

`tablei` er en boks for tabelloppslag; overføringsfunksjonen er angitt i p6. Siden `aosc` i utgangspunktet varierer mellom -250 og +250, har vi brukt en *variabeltilordning* for å addere 256. På den måten får vi en peker inn i tabellen, som er 513 samples lang.

```
;waveshaping instrument

sr=32000
kr=3200
ksmps=10
nchnls=1

instr   1

kenv    linen   p4,.1,p3,.1
aenv    expseg  1,p3*p7,250,p3*p8,150,p3*p9,p10,p3*.5,.1
aosc    oscili  aenv,p5,1
aosc    =       aosc+256
atab    tablei  aosc,p6
        out     atab*kenv
endin
```

I partiturfilen har vi brukt GEN08 for å fylle tabellen. GEN08 lager en glatt, fin interpolering (spline) mellom de oppgitte punktene.

```
f1 0 2048 10 1
f2 0 513 8 -1 89 -.5 89 -.6 156 .6 89 .5 89 1
f3 0 513 8 -1 100 0 100 -.4 112 .4 100 0 100 1

i1    0     6     10000    275    3    .1     .05    .3   20
i1    0     6      5000    350    3    .1     .05    .3   20
i1    0     6      3300    425    3    .1     .05    .3   20
i1    0     9      2500    475    2    .06    .04    .4   70
i1    0     9      2000    525    2    .06    .04    .4   70
i1    0     9      1600    600    2    .06    .04    .4   70
i1    0     9      1400    675    2    .06    .04    .4   70
```

4.5 FOF

Det finnes en egen boks for FOF-syntese. fof er en ganske kompleks enhetsgenerator, og den har fått et helt appendix i Csound-manualen. Her er et instrument man kan prøve:

```
      sr = 32000
      kr = 2000
      ksmps = 160

      instr    1
k1    oscil    2, 4, 1
k2    linseg   0, p3/2, 0, p3/2, 0.8
k3    expseg   1, p3/3, 1, p3, 0.001
a1    fof      5000, p4+k1,  500*k3, k2, 120, .003, .017, .005, 10, 1, 2, p3
a2    fof      4000, p4+k1,  850*k3, k2, 120, .003, .017, .005, 10, 1, 2, p3
a3    fof      3000, p4+k1, 2500*k3, k2, 120, .003, .017, .005, 10, 1, 2, p3
a4    fof      2000, p4+k1, 3500*k3, k2, 120, .003, .017, .005, 10, 1, 2, p3
a5    =        a1 + a2 + a3 + a4
      out      a5
      endin

f1 0 4096  9 1 1 0
f2 0 1024 19 .5 .5 270 .5
i1 0 12 100
i1 1 11 133
i1 2 10 150
e
```

4.6 Karplus-Strong

Karplus-Strong-algoritmen har også en egen boks, som heter `pluck`. Se Csound-manualen for detaljene, men `pluck` er enkel å bruke:

```
      sr = 32000
      kr = 200
      ksmps = 160

      instr    1
a1    pluck    ampdb(p5),cpspch(p4),cpspch(p4),0,1
a2    linen    a1,.01,p3,p3*0.3
      out      a2
      endin
```

I partiturfilen bruker vi en ny fidus, nemlig *tempo*-kommandoen `t` som justerer alle tidspunkter og varigheter. `t 0 90` betyr at tempo skal settes til 90 beats pr. sek. istedenfor de vanlige 60, fra tidspunkt 0.

```
t 0 90
i1 0   0.25 6.04 70
i1 0.6 .    6.07
i1 1.2 0.9  6.09
i1 +   0.25 6.04
i1 2.7 .    6.07
i1 3.3 0.3  6.10
i1 +   1.0  6.09
i1 4.8 0.25 6.04
i1 5.4 .    6.07
i1 6.0 0.9  6.09
i1 +   .    6.07 75
i1 7.8 4.   6.04 70
i1 .   .    5.04
i1 .   .    5.09
e
```

Kapittel 5

Lydbehandling

Det finnes bokser i Csound som leser fra forhåndsproduserte lydfiler. Dette gjør at Csound er like velegnet til lydbehandling som til lydsyntese.

5.1 Kor

I instrumentet under bruker vi `soundin`, som leser en angitt lydfil. Nytt er også `delayw` som skriver inn i en delay, `delayr` som angir en fast delaytid, og `deltapi` som er en variabel tapning av delayen. Tre tapninger beveger seg her langsomt fram og tilbake, ute av takt med hverandre:

```
    sr=32000
    kr=32000
    ksmps=1

instr 1
ainnlyd   soundin   "Oyvind/testlyd"
adummy    delayr    1
amod1     oscili    .05,.1,1
atap1     deltapi   amod1+.05
amod2     oscili    .05,.11,1
atap2     deltapi   amod2+.05
amod3     oscili    .05,.12,1
atap3     deltapi   amod3+.05
          delayw    ainnlyd
          out       (atap1+atap2+atap3)/3;
endin

f1 0 8192 10 1
i1 0 45
e
```

5.2 Global romklang

Ofte vil man legge en effekt på mange instrumenter samtidig, eller man vil la en varierende kontroll affisere et antall instrumenter. For å oppnå dette, bruker man *globale variable*. Globale variable begynner med bokstaven *g*, etterfulgt av en bokstav etter det vanlige systemet (a, k, i). Det er lurt å initialisere globale variable i orkester-headeren, med kommandoen `init`.

Eksemplet under viser hvordan romklangen kan kjøre helt uavhengig av andre instrumenter. Etter at instrumentene har sendt sine outputs til den globale variabelen, kan den brukes som input i det separate romklangsinstrumentet. Siden instrumentene adderer inn i den globale variabelen, må vi også passe på å nulle den etter at den er brukt i romklangsinstrumentet (dette kan kanskje virke litt pussig).

```
    sr    = 32000
    kr    = 1000
    ksmps = 32
    garev init 0

instr 1                        ; Hvitstoy
  anois rand    5000
  garev =       garev+anois    ; Send til reverb
        out     anois          ; og direkte ut
endin

instr 2                        ; 220 Hz pip
  abeep oscil   10000,220,1
  garev =       garev+abeep
        out     abeep
endin

instr 99                       ; Romklang
  asig  reverb  garev*.1,2     ; Input fra garev, 2 sek. klangtid.
  out   asig
  garev =       0
endin


f1  0  4096  10  1
i99 0  5                       ; Skru paa klangen i 5 sek.
i1  0  .1
i2  1  .1
i1  2  .1
i2  2  .1
e
```

5.3 Fasevokoding

Det separate programmet `pvanal` kan gjøre en fasevokodingsanalyse av en lyd. Analysedataene sendes til en egen fil, som siden kan leses inn i et Csound-instrument for resyntese med enhetsgeneratoren `pvoc`.

`pvanal` krever en mono AIFF-fil (ikke AIFC), og programmet kjøres med UNIX-kommandoen "`pvanal innfil analysefil`". Innfilen blir søkt etter under `/sounds` eller hva man ellers har satt i systemvariabelen `SFDIR`.

`pvoc` leser fra analysefilen med en varierende *peker*. Denne pekeren kan bevege seg hurtig eller langsomt gjennom lyden, gå baklengs etc. uten at pitch endres. På denne måten kan man f.eks. lage time-stretch. I eksemplet strekkes lyden med en faktor 2, samtidig som vi transponerer en kvint opp:

```
sr = 44100
kr = 441
ksmps = 100

instr  1
ktime   line   0, p3, p3/2
aout    pvoc   ktime, 1.5, "pvfil"
out    aout
endin

i1 0.0 30.0
e
```

5.4 Ceres

Ceres er et annet fasevokodingsprogram med en kobling mot Csound. Ceres er menystyrt, og relativt enkelt å bruke. Med *Sieve*-funksjonen kan man plukke ut bare de sterkeste deltonene på ethvert tidspunkt, og disse små "ploppene" kan betraktes som egne noter. Ceres kan skrive en Csound partiturfil med alle disse notene.

5.5 Lineær-prediktiv koding

Det separate programmet `lpcanal` kan gjøre en lineær-prediktiv analyse av en lyd. Analysedataene sendes til en egen fil, som siden kan leses inn i et Csound-instrument for resyntese med enhetsgeneratoren `lpread`.

`lpcanal` krever en mono IRCAM-fil. Man må derfor først konvertere til dette formatet, f.eks. med programmet `sox`. Kommandoen "`sox innfil.aiff utfil.sf`" skulle gjøre jobben (suffixene må være som angitt for at sox skal gjenkjenne formatene).

`lpcanal` kjøres så med UNIX-kommandoen "`lpcanal innfil analysefil`". Med på lasset følger en (upålitelig) pitch-tracker og en RMS-analyse. Se Csound-manualen for detaljene. Analysefilen leses inn i Csound f.eks. slik:

```
sr = 32000
kr = 1000
ksmps = 32

instr   1
ktime                  line     0, p3, p3
krmsr,krmso,kerr,kcps  lpread   ktime, "analyse.lpc"
kcps                   =        (kcps == 0 ? 220 : kcps)
avoice                 buzz     krmso, kcps, int(sr/880), 1
aunvoc                 rand     krmso
asig                   =        (kerr < .1 ? avoice : aunvoc)
aout                   lpreson  asig
                       out      aout
endin
```

Her var det mye rart! For det første bruker `lpread` en tidspeker, her kalt `ktime`, akkurat som `pvoc`. Slik kan man lage hastighetsendringer etc. `lpread` oppdaterer variablene `krmsr` (RMS av residualen, brukes sjelden), `krmso` (RMS av originalsignalet), `kerr` (andelen støy), og `kcps` (pitch-tracking). Dessuten oppdateres alle filterkoeffisientene som senere brukes av filteret `lpreson`.

I tilordningene brukes en syntaks som er kjent fra programmeringsspråket C. Den første tilordningen skal tolkes slik: "Hvis `kcps` er 0 så sett `kcps` til standardfrekvensen 220, ellers sett den tilbake til `kcps`". Den andre tilordningen tester på `kerr` for å bestemme om eksitasjonssignalet skal tas fra `buzz` eller `rand`. Dette tilsvarer omtrent en `switch` i Max.

LPC er en kraftig og morsom teknikk. Prøv f.eks. å la pitch være konstant (robotstemme), eller la pitch styres fra RMS-verdiene! Effekter som "snakkende orkester" etc. er også mulige. Ha imidlertid ikke for store forhåpninger til lydkvaliteten når du bruker LPC.

5.6 Deltoneanalyse/additiv resyntese

Additiv syntese kan gjøres med vanlige oscillatorer og kontrollfunksjoner som `line`, men det finnes en mer praktisk metode for komplekse tilfeller. Enhetsgeneratoren `adsyn` leser en egen kontrollfil, der amplityde og frekvens til opptil 50 deltoner spesifiseres til diskrete tidspunkter. `adsyn` interpolerer lineært mellom disse. Et eget grafisk brukergrensesnitt eksisterer for å editere `adsyn`-kontrollfiler. Start dette programmet med UNIX-kommandoen `adsyn`.

Kontrollfilen kan senere leses inn i et Csound-instrument, slik:

```
sr=32000
kr=32
```

```
ksmps=1000

instr 1
aout  adsyn  1,1,1,"test.ads"
      out aout/2
endin
```

Divisjonen med 2 er bare for å unngå klipping. Man må kanskje justere litt her. Det er en feil i `adsyn` som gjør at den er ganske hårsår når det gjelder kontrollraten. `ksmps=1000` later til å fungere.

Amplityder, frekvenser og tidspunkter kan skaleres med de tre første parametrene til `adsyn` (her er de alle skalert med identiteten).

Det separate programmet `hetro` kan gjøre en filterbankanalyse av en lyd der amplityde og frekvens til et antall deltoner spores og lagres i en `adsyn` kontrollfil. `hetro` krever en mono AIFF-fil (ikke AIFC), og programmet kjøres med UNIX-kommandoen "`hetro innfil analysefil`". Det er endel nyttige flagg til denne kommandoen, som er beskrevet i Csound-manualen. `hetro` er beryktet for sine klikk og annen støy.

5.7 Videre

Nå skulle du være i stand til å fortsette utforskingen av Csound på egenhånd, ved hjelp av Csound-manualen. Gå gjerne gjennom "tutorialene" til Richard Boulanger, appendix 2 i manualen. Filene finnes i `/home/notam/b/csound/tutorfiles`.