Implementing “whistler” sound by using sound synthesis

Physics 406

Oral Presentation Project

Dongryul Lee

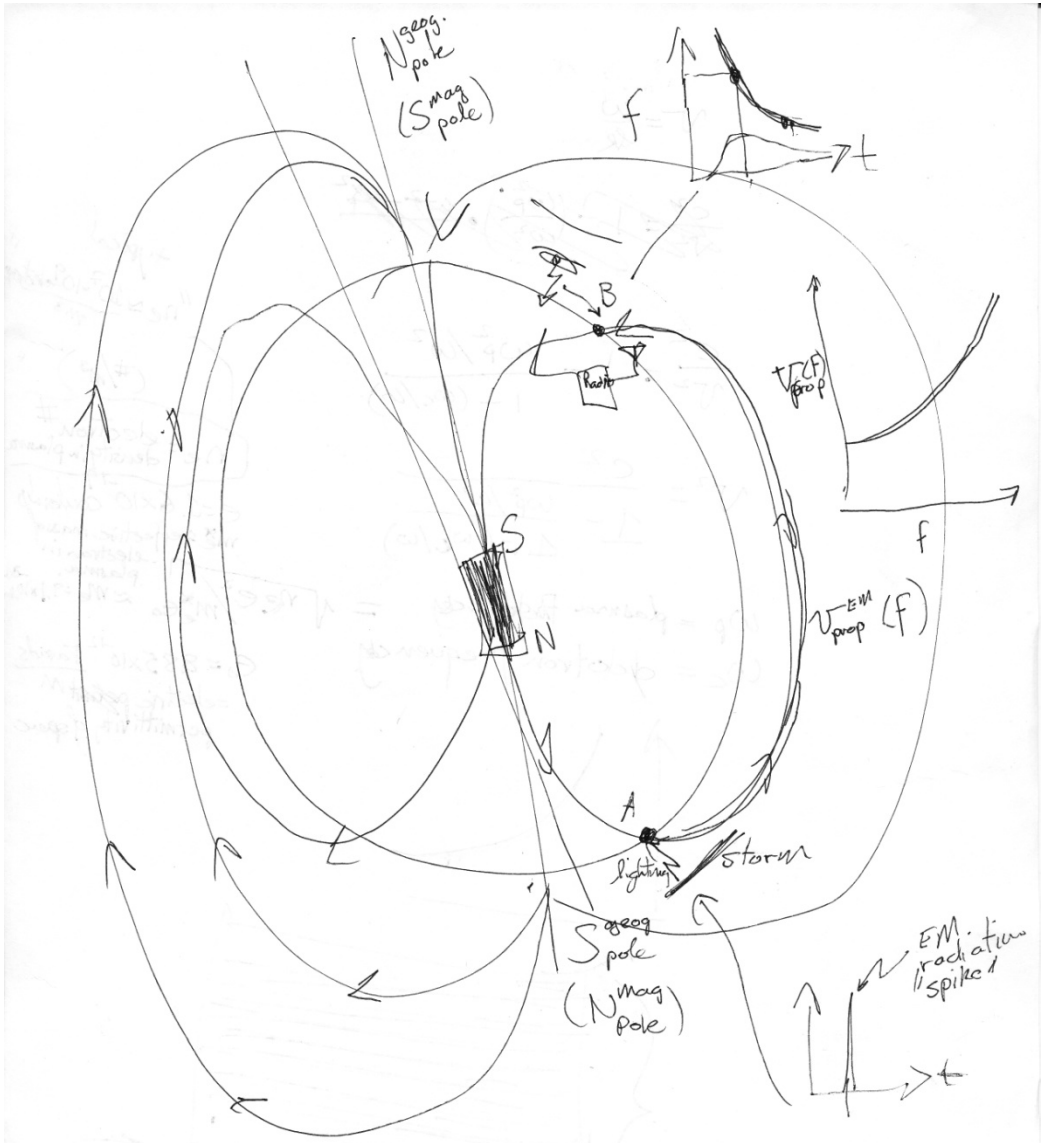
Whistler sounds

- "Encounters at the End of the World"

  http://www.youtube.com/watch?v=OlrcbKlW4Tw (from 1' 25")

- Is this really seal calling?

- What Prof. Errede Said:



whistler dispersion.

→ waves in plasma

Additive and Subtractive synthesis

- Additive synthesis: a class of sound synthesis techniques based on the summation of elementary waveforms to create a more complex waveform. (Roads, the computer music tutorial)
  - Addition of Partials
  - Changing Phase Factors
  - FM (Frequency Modulation)

- Subtractive synthesis: sound synthesis using filters to shape the spectrum of a source sound. (Ibid)
  - Delay, Chorus
  - Bandpass filters

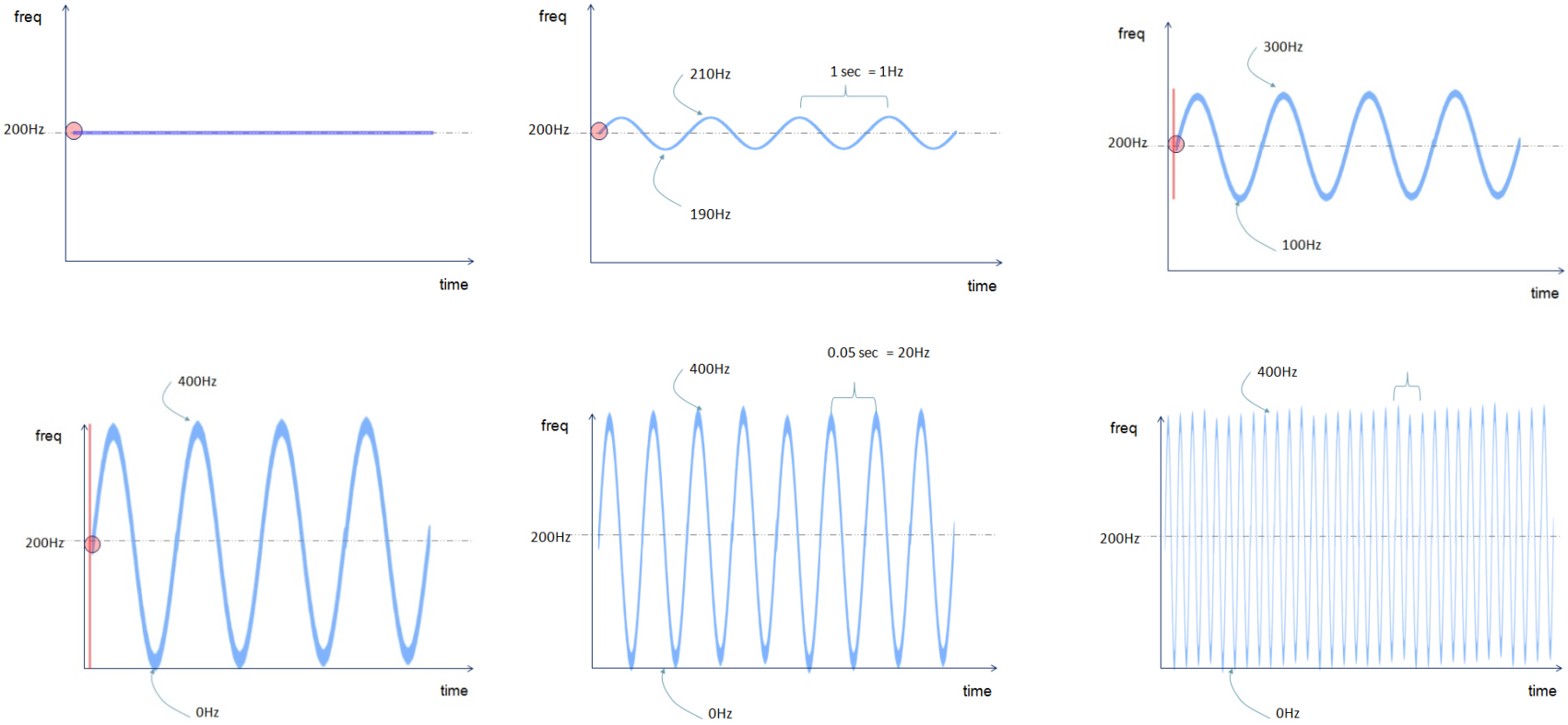
Synthesizer / Frequency Modulation

- KORG TR-rack (1998)

  - A Sound module without the master keyboard
  - I used for 3 years
  - "Whistler" Sound in the sound banks

    - A126 *The UFO Appears*

FM (Frequency Modulation), Chowning



From Chowning, John Lecture at the Eastman School of Music, 2-15, 2012
used for in-class oral presentation for educational purposes

FM (Frequency Modulation), Chowning

The equation for a frequency-modulated wave of peak amplitude A where both the carrier and modulating waves are sinusoids is

$$e = A \sin(\alpha t + I \sin\beta t) \tag{1}$$

where

e = the instantaneous amplitude of the modulated carrier
α = the carrier frequency in rad/s
β = the modulating frequency in rad/s
$I = d/m$ = the modulation index, the ratio of the peak deviation to the modulating frequency.

All of the above relationships are expressed in the trigonometric expansion of Eq. [2]

$$\begin{aligned} e = A\{ \; & J_0(I) \sin\alpha t \\ & + J_1(I)\,[\sin(\alpha+\beta)t - \sin(\alpha-\beta)] \\ & + J_2(I)\,[\sin(\alpha+2\beta)t + \sin(\alpha-2\beta)] \\ & + J_3(I)\,[\sin(\alpha+3\beta)t - \sin(\alpha-3\beta)] \\ & + \ldots\ldots\ldots\ldots\ldots \}. \end{aligned} \tag{2}$$

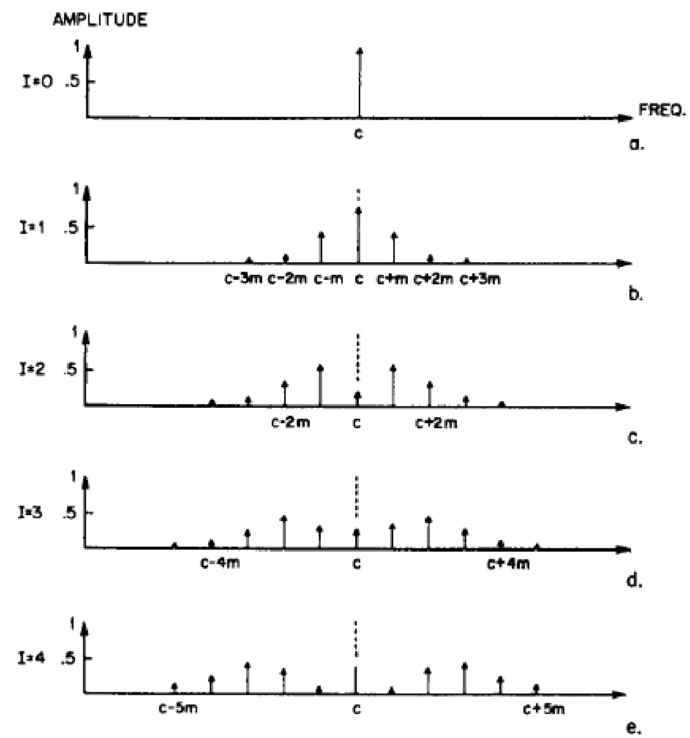
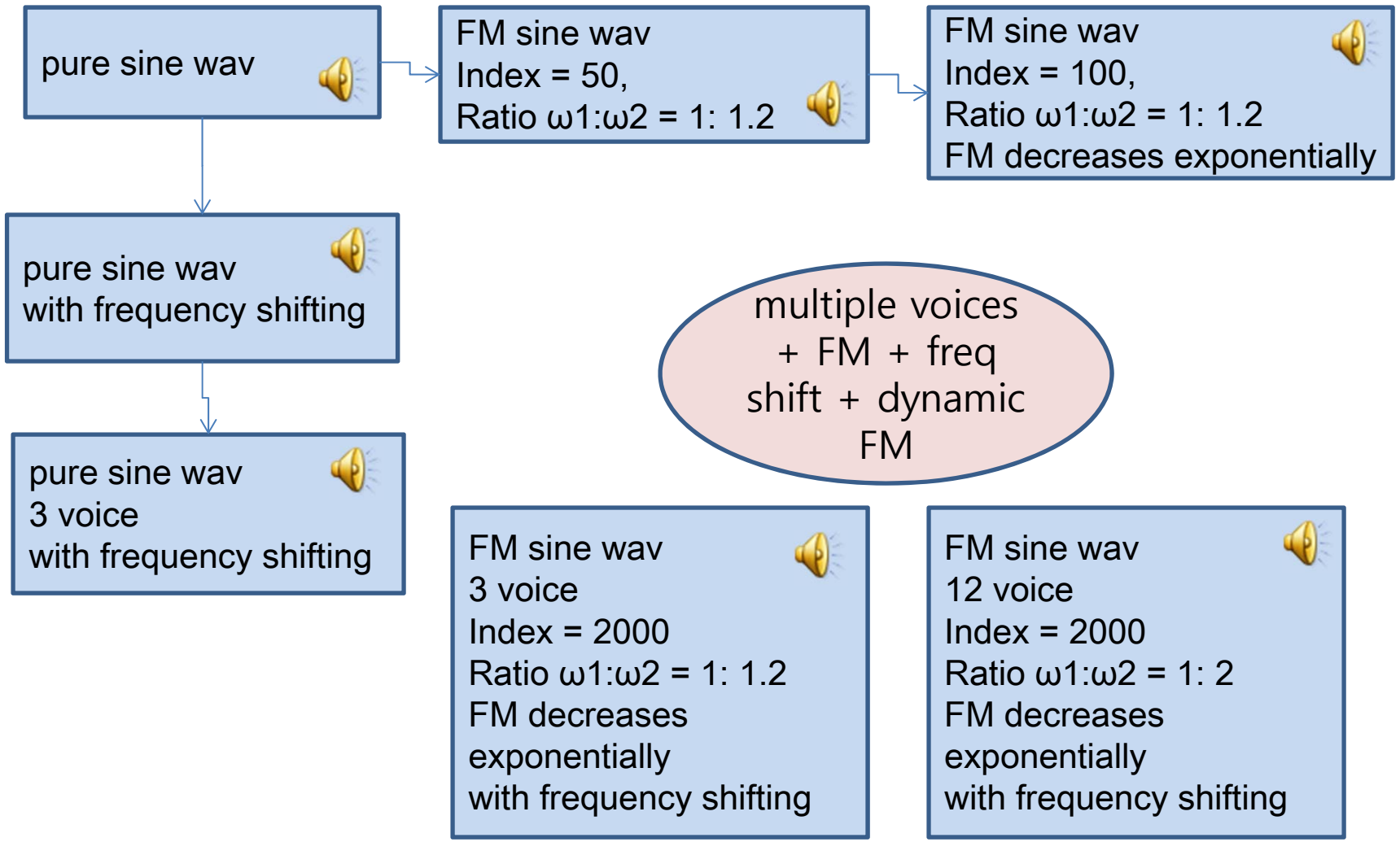


Fig. 1. Example to show increasing bandwidth with increasing modulation index, I. The upper and lower side frequencies are at intervals of the modulating frequency, m, and are symmetrical around the carrier, c.

The Synthesis of Complex Audio Spectra by Means of Frequency Modulation, Chowning, J. Audio Eng. Soc. 21, 7, 1973

Whistler Implementation using CSound



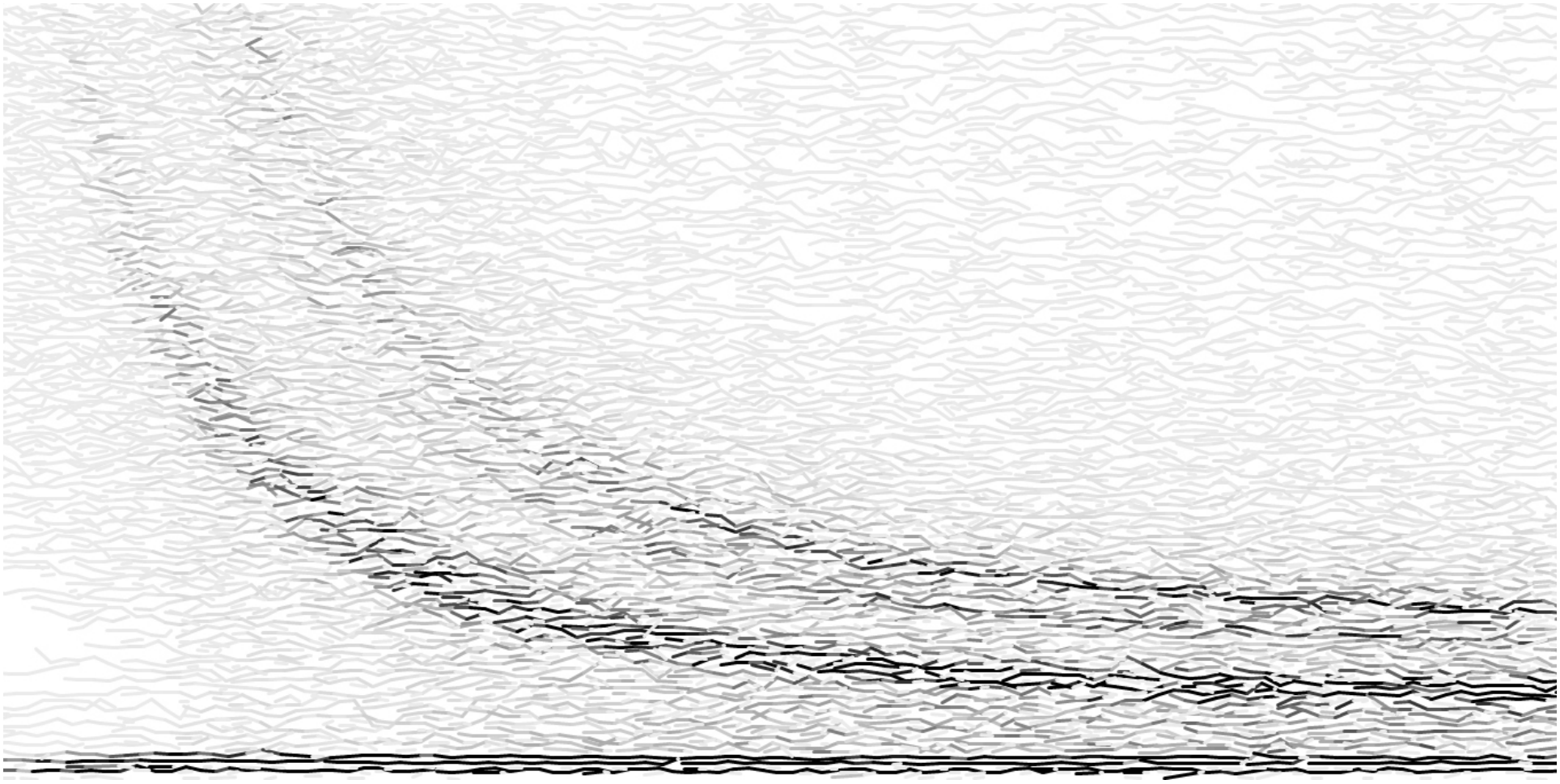
Csound source code (excerpt)

```
sr = 96000kr  =  960ksmps= 100nchnls = 2instr 1; ************************************************************** idur  = p3 ;  p3 = duration  ifundpitch  =
cpspch(p4) ; p4 = fundamental frq or pch (a4 = 440hz = 8.09 pch)  ifundpitch = ifundpitch / 2 ;octave transpose  ;ifundpitch = p4;  ;print ifundpitch;  iamp
= p5;  ;print ifundfrq  iharmratio = p6; p6 = harmonic ratio  ifrqtable = p7; p7 = basic frq shape table  iampenvtable = p8; p8 = basic amplitude table
iamprepeat = p9; p9 = amplitude table repetition  iovtratiotable = p10; p10 = ovt table (-1 is no ovt)  iovtcount = p11; p11 = # of ovts  iovtamptable = p12;
p12 = amp ratio of ovts  ileft = sqrt(p13);p13 = panning  iright = sqrt(1.-p13);      ;glissando  iglissratio = p14; p14 = glissando ratio (final pitch /
fundamental pitch)    ;FM  iModularFrqRto = p21 ; Modular frq / Carrier frq  ;max amplitude;  ;imaxamp = 5000;  ;temporary : high ovt -> low amp, low ovt
-> high amp      if (iharmratio >= 60) then          imaxamp = 1000  elseif (iharmratio >= 40) then          imaxamp = 2000  elseif (iharmratio >= 30) then
            imaxamp = 3000  elseif (iharmratio >= 20) then          imaxamp = 4000  else                imaxamp = 5000  endif  irealamp = iamp *
imaxamp;  ;calculate fundamental frq  ifundfrq = ifundpitch * iharmratio  ;print ifundfrq;  ;create amplitude shape  ;if iamprepeat=1 then sustain the env for
the duration of the note  if (iampenvtable != -1) then ;iampenvtable=p8                aenv                                oscil3 irealamp,
(1/idur)*iamprepeat, iampenvtable  else                                 ;generate amplitude env                             ;use p31 - p40 fields
                            ;p31: peak portion of the env. (temporary)                                                      ;can not use iamprepeat feature!!! (p9)
                              ipeaktime = p31 * idur              iremainingtime = (1 - p31) * idur       ;lets add some more files linseg, expseg...
             aenv                                linseg              0.00001, ipeaktime, irealamp, iremainingtime, 0  endif  ;double envelope
shape  ;added feature after mist_solo_2b  idblenv = p41  idblenvrepeat = p43  idblenvintensity = p44  if (idblenv == -2 ) then ; create real time env
            idblpeaktime = p42 * idur       idblremainingtime = (1 - p42) * idur         adblenv          linseg              0.00001, idblpeaktime, 1,
idblremainingtime, 0.00001    aenv = aenv * (adblenv^idblenvintensity)  elseif (idblenv != -1) then ; function table dbl env adblenv  oscili 1,
idblenvrepeat/idur, idblenv     ;aenv = aenv * (adblenv^idblenvintensity)              adblenvpow pow adblenv, idblenvintensity              aenv product
aenv, adblenvpow  endif  ;pitch vibration  ipitchvib = p46  if (ipitchvib != -1) then      ipitchvibrepeat = p47               ipitchvibintensity = p48
            afreqvib          oscil3            1, ipitchvibrepeat/idur, ipitchvib  else          afreqvib = 1 ;no vibration  endif    afreqvib =
afreqvib ^ ipitchvibintensity  ;glissando  ;no gliss  if (iglissratio == -1) then ;create fundamental note sound                aout
            oscil3 aenv, ifundfrq * afreqvib , ifrqtable   ;with glissando  else                                            itargetfrq = ifundfrq * iglissratio
             ; line changed to expon              kfrq expon ifundfrq, idur, itargetfrq                aout                                oscil3 aenv,
kfrq * afreqvib , ifrqtable  endif  ;Frequency Modulation  if (iModularFrqRto != -1) then          iModularFrq = ifundfrq * iModularFrqRto
            iIndexRto = p22   iIndexEnvFuncTbl = p23          iFMrepeat = p24;  iModularFrqOscilTbl = ifrqtable ; use same oscil function tbl as fund
            aFMEnv            oscil3              iIndexRto, iFMrepeat/idur , iIndexEnvFuncTbl                        ;Modular frq is not changing
            aMod              oscil3            aFMEnv,           iModularFrq, iModularFrqOscilTbl    ;Modular frq is changing (by gliss.)   ;aMod
            oscil3  aFMEnv, kfrq * iModularFrqRto, iModularFrqOscilTbl                                  if (iglissratio == -1) then
            aout              oscil3            aenv, ifundfrq * afreqvib + aMod, ifrqtable           else                ;gliss with FM
            aout              oscil3            aenv, kfrq * afreqvib + aMod, ifrqtable               endif    endif  ;use loop_ge and create overtones
from ovtratiotable  ;lood (aout = aout + generatedovt), while i < # of ovt  out ileft*aout, iright*aout;endin
```

Analyzing harmonic contents by using FFT

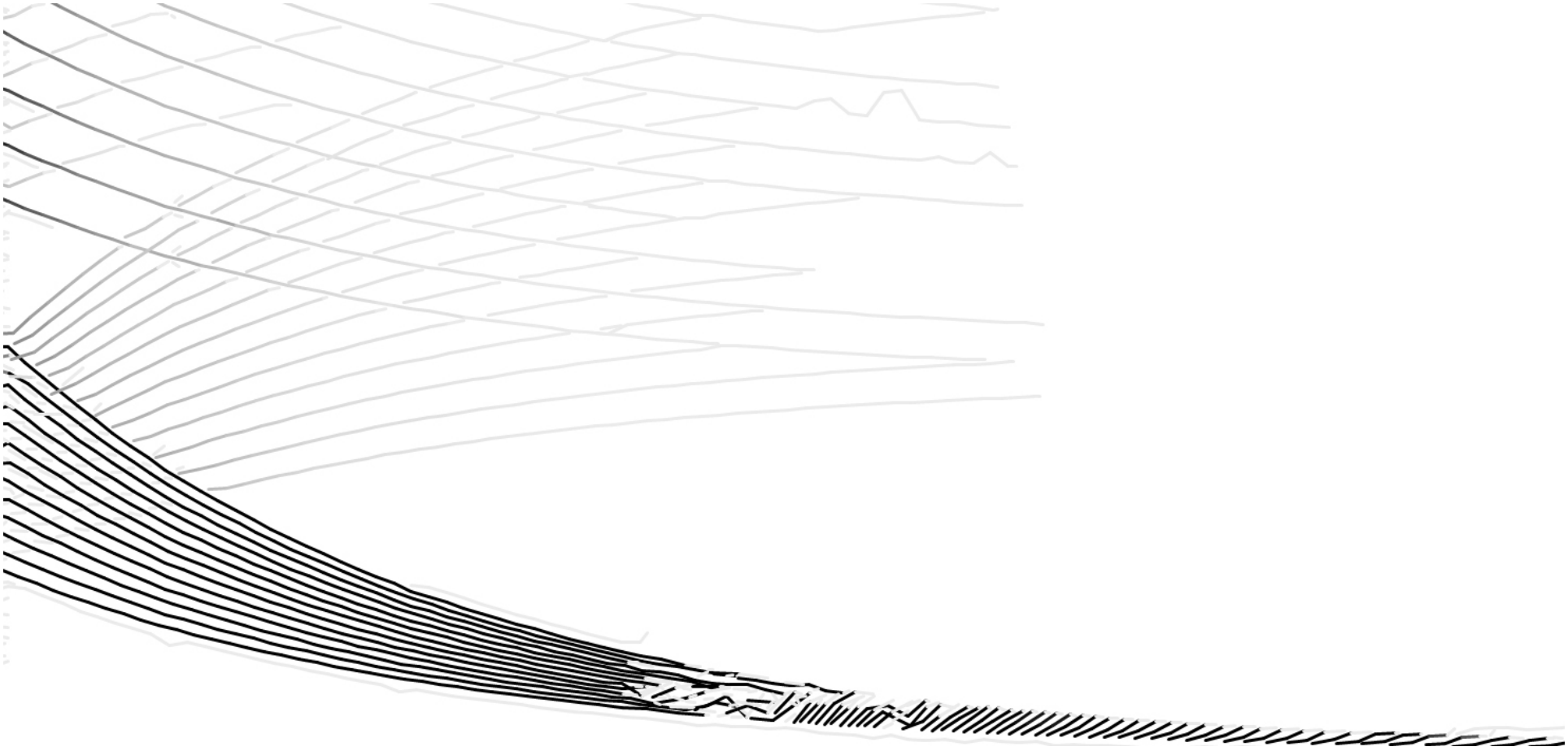
original whistler sound

Analyzing harmonic contents by using FFT



TR-rack *UFO Appears* sound

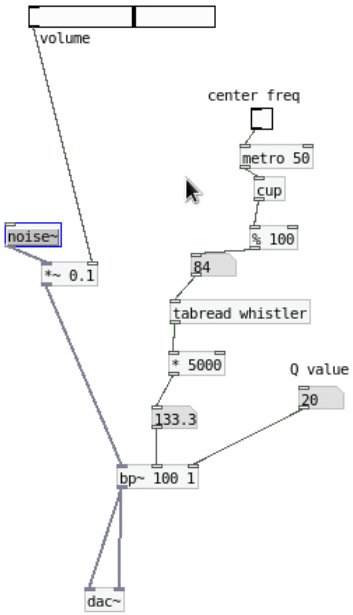
Analyzing harmonic contents by using FFT

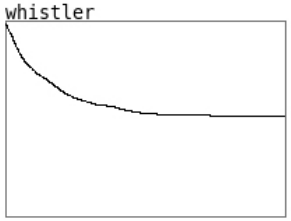


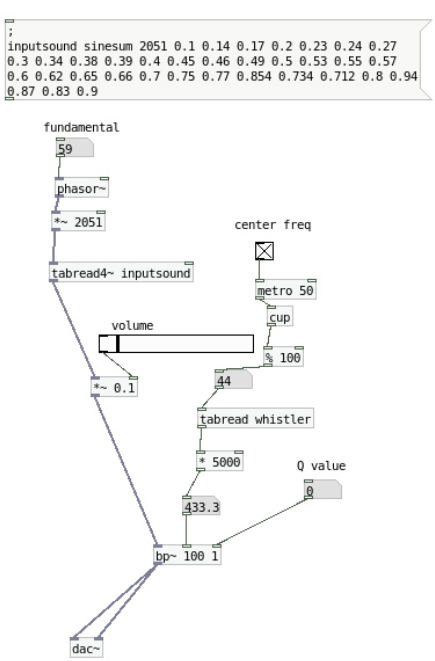
Csound FM whistler sound

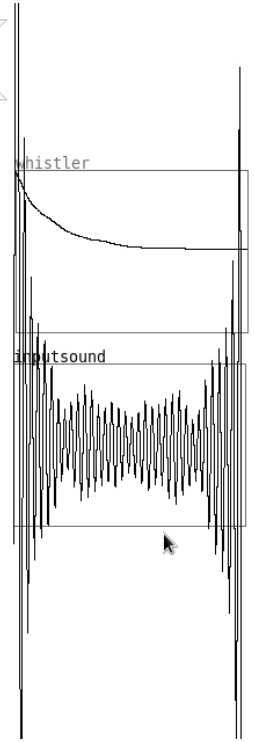
Another approach: subtractive synthesis

- Implementing “thunder sound” by using PureData
  - Using bandpass filters, multiple sound source, an envelope
- bandpass filter with exponentially decreasing frequency center









Sound sources: white noise, static noise, sine wave

white noise

static noise

PD Implemented whistler simulation

white noise
with bandpass filter

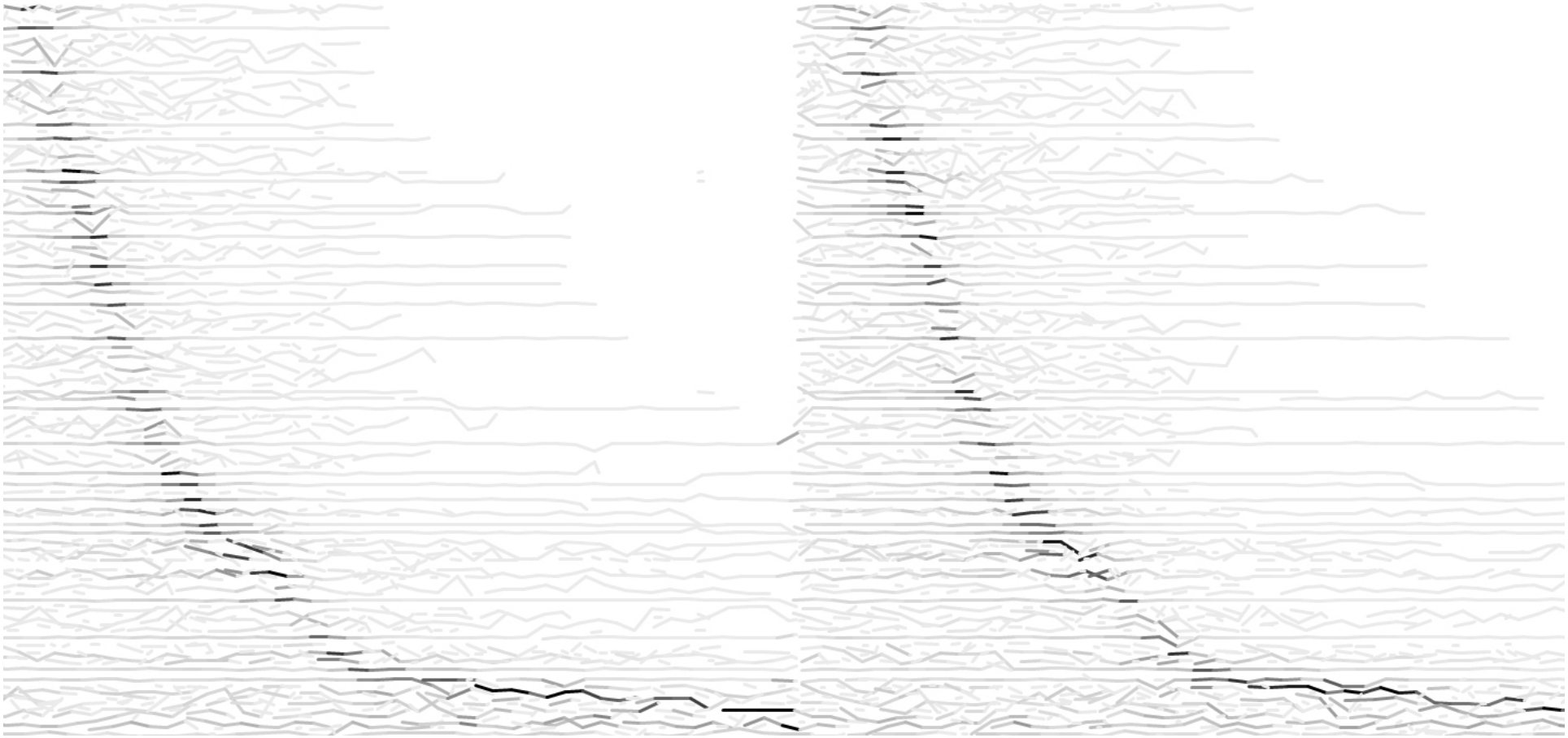
static noise
with bandpass filter,
Q=5

sawtooth sinewave
with bandpass filter

static noise
with bandpass filter,
Q=25

white noise + static noise
with bandpass filter, Q=25

Analyzing harmonic contents by using FFT



PD generated whistler sound