Glitch Free FM Vocal Synthesis

Chris Chafe
Center for Computer Research in Music and Acoustics, Stanford University
cc@ccrma.stanford.edu

ABSTRACT

Frequency Modulation (FM) and other audio rate non-linear modulation techniques like Waveshaping Digital Synthesis, Amplitude Modulation (AM) and their variants are well-known techniques for generating complex sound spectra. Kleimola [1] provides a comprehensive and up-to-date description of the entire family. One shared trait is that synthesizing vocal sounds and other harmonically-structured sounds comprised of formants can be problematic because of an obstacle which causes distortions when intensifying time-varying controls.

Large deflections of pitch or phoneme parameters cause jumps in the required integer approximations of formant center frequencies. Trying to imitate human vocal behavior with its often wide prosodic and expressive excursions causes audible clicks. A partial solution lay buried in some code from the 80's. This, combined with a phase-syncronous oscillator bank described in Lazzarini and Timoney [2] produces uniform harmonic components which ensure artifact-free, exact formant spectra even under the most extreme dynamic conditions. The paper revisits singing and speech synthesis using the classic FM single modulator / multiple-carrier structure pioneered by Chowning [3]. The revised method is implemented in Faust and is as efficient as its predecessor technique. Dynamic controls arrive multiplexed via an audio rate "articulation stream" which interfaces conveniently with sample-synchronous algorithms written in Chuck. FM for singing synthesis can now be "abused" with radical time-varying controls. It also has potential as an efficient means for low-bandwidth analysis – resynthesis speech coding. Applications of the technique for sonification and in concert music are described.

1. INTRODUCTION

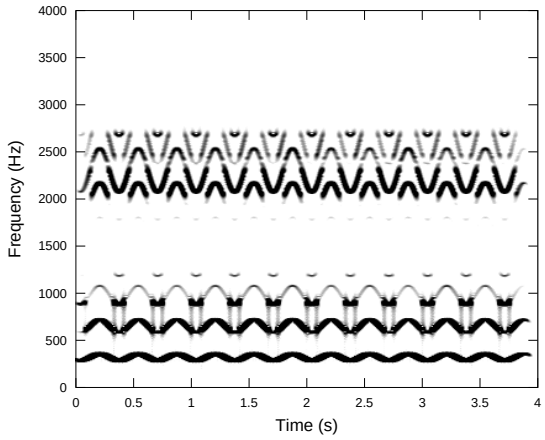
Synthesis of singing voice by computer has a history which begins in the very first years of computer music. The song *Daisy Bell (Bicycle Built for Two)* was sung by a computer in 1961 in an arrangement by Max Mathews and Joan Miller with vocal synthesis by John Kelly and Carol Lochbaum when the Bell Telephone Laboratories experiments with digital music synthesis were only 4 years old. It was an early case of analysis – resynthesis speech coding providing a means for singing synthesis. Over the decades, most synthesis technique have been applied to emulate the singing voice (additive, subtractive, physical model, FOF, etc.). The quest continues more than fifty years later with composers attracted to vocal synthesizers like Yamaha's Vocaloid [1] where they can explore a fascination with musical personalities of singers which never existed. This paper joins a thread which began with Chowning's work in the late 70's, early 80's involving FM for vocal synthesis and which has been virtually languishing since it's early use in a few musical works.

John Chowning's FM singing voice method was first described in his 1980 article [3] prior to completing *Phonē* at IRCAM (1981). The multi-channel tape piece features a wide variety of singing voices and morphing of vocal timbres with other FM-generated timbres such as gongs. The technique creates multiple formants with independent tunings using multiple carriers and a shared modulator. Two formants are used for his version of a soprano voice "eee" and three formants for his spectrally-rich basso *profondissimo*. A later version adds a third formant to the soprano model in a synthesis of the vowel "ahh" [4]. Pitch vibrato which causes synchronous spectral modulation is especially effective and Chowning has often demonstrated how crucial this is to rendering vowels convincingly. "It is striking that the tone only fuses and becomes a unitary percept with the addition of the pitch fluctuations, thus *spectral envelope does not make a voice!*" [3].

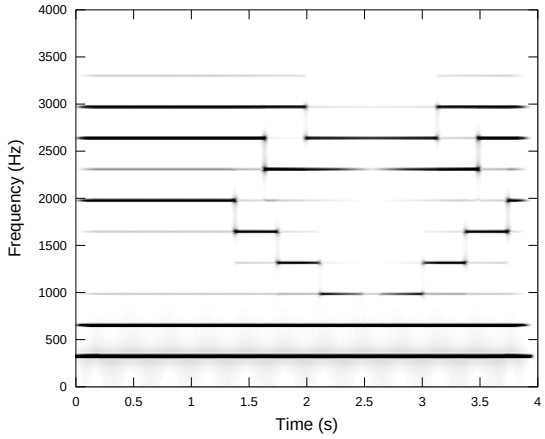
The method has an inherent shortcoming which limits the amount of vibrato excursion and limits phoneme transitions to nearby phonemes. Beyond these limits noticeable artifacts occur which are caused by discrete shifts of formant center frequency. Discontinuities are perceived as clicks and result from integer shifts in the carrier to modulator ratio $c : m$ which are required in order to track a desired formant center frequency f_c for a given pitch f_p. The modulating oscillator is always set to f_p, so $m = 1.0$. The carrier ratio c is an *integer approximation* and quantization of the actual *real* ratio f_c/f_p.

Formant synthesis with FM is essentially contradictory to the physics. The harmonic nature of voiced sound allows only harmonic number ratios $c \in \mathbb{N}_{\geq 1}$ for the carrier. Where physical sound production is an excitation – resonance mechanism with independent tuning of both elements, FM models can only approximate the resonance frequencies of the latter when constrained to produce harmonic spectra. The inherent problem is that these approximations are discontinuous in frequency. In practice, this

[1] Vocaloid3 uses a triphone frequency-domain concatenative synthesis engine.

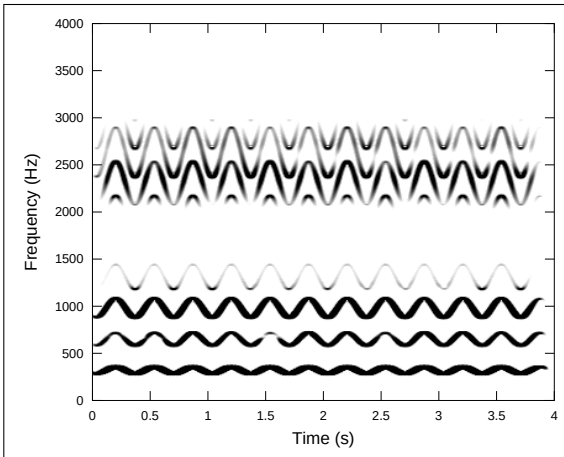


(a) vibrato

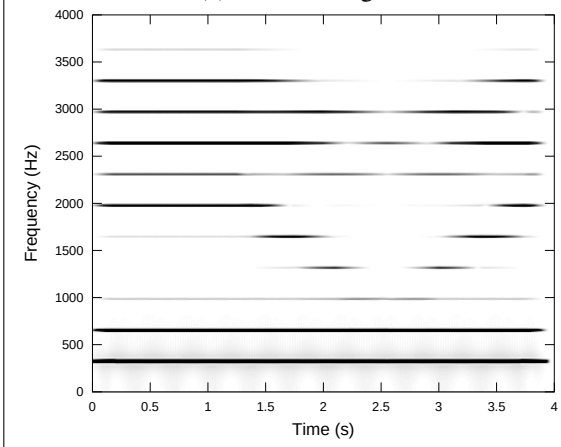


(b) vowel alternation "eee–ooo–eee"

Figure 1. Clicks always occur when transitions to a new formant center frequency f_c forces a carrier oscillator to change its harmonic ratio. FM vocal formants use a $c : m$ ratio where $c \in \mathbb{N}_{\geq 1}$ and $m = f_p$, where f_p is the desired pitch.



(a) vibrato – deglitched



(b) vowel alternation – deglitched

Figure 2. Result of applying the solution adopted from Le Brun to the synthesis shown in Fig.1.

severely limits the amount of pitch skew not only constraining vibrato, but also portamento and glissando to small ranges. In the method's original form, it is impossible to shift ratios without causing glitches like those shown in the spectrograms of Fig. 1.

Finding a solution became necessary in order to use FM as the synthesis engine for a sonfication project involving brain signals. FM singing offers advantages for this body of work which attempts to fashion a singing choir direct from the mind. The goal isn't what one probably first imagines e.g., an ensemble of mind-controlled voices. Instead, this is a technology for auditory display of the rapid fluctuations of EEG and electrocorticography (eCog) recordings. Singing voice synthesis has its attractions in that it can allude to imagery of "inner voices" but it is also particularly apt because of the ease with which listeners lock on to patterns of phonemic and other voice-like timbral transitions. The range of data encountered in brain recordings (from quiescent to seizure) and a desire to have a very flexible mapping strategy have been motivations for the present investigation into solving the discontinuity problem. The completed work will reach the public as a gallery installation (exploring recorded data) and as a medical monitoring device for detecting seizures (with the singing voice controlled directly from electrodes in real time).

2. EARLY SOLUTION

Marc Le Brun described digital waveshaping synthesis in 1979 as a generalized paradigm for non-linear modulation synthesis [5]. FM is a special case of waveshaping synthesis and in devising a way to avoid the discontinuity problem for waveshaping, Le Brun also solved it for the FM case. Le Brun's solution remains unpublished (until now) with one exception: Bill Schottstaedt has preserved it as a synthesis instrument in the Common Lisp Music (CLM) project [6]. From the code comment, "**Vox**, an elaborate multi-carrier FM instrument is the voice instrument written by Marc Le Brun, used in *Colony* and other pieces."

Vox avoids the integer ratio shift discontinuities by implementing a cross-fading solution. Two carriers corresponding to even and odd harmonic numbers are assigned

to each formant "bracketing" the true formant center frequency. Their assignments are made from the two nearest harmonics $f_{lower} = \lfloor f_c/f_p \rfloor$ while the other is the nearest upper harmonic $f_{upper} = \lceil f_c/f_p \rceil$. The assignment of harmonics to individual oscillators is dynamic and depends on whether they are even numbered or odd numbered. When an oscillator is required to change its harmonic number the other will be approaching the actual target f_c/f_p. The two carrier oscillators' amplitudes sum to unity in a mixture whose gains are complementary and linearly determined by proximity to the target. The key feature which makes this work is that it ensures that the oscillator which is having its frequency changed will be muted. As a nice side-effect, it also sharpens the accuracy with which the target formant center frequency is being synthesized.

Le Brun's paper describes "a unified conceptual framework for a number of nonlinear techniques, including frequency-modulation synthesis. Both the theory and practice of the method are developed fairly extensively, beginning with simple but useful forms and proceeding to more complex and richer variations." The cross-fade solution however only existed in code from the same era. To detail the historical record precisely, its first implementation was written in the MUS10 compiler (Stanford Artificial Intelligence Laboratory's version of Bell Laboratories' MusicN compilers). Later, it was ported to CLM as **pqw-vox** a "translation from MUS10 of MLB's waveshaping voice instrument (using phase quadrature waveshaping)." Today, both **pqw-vox** and the FM version **vox** can be found translated to Scheme in Schottstaedt's Snd project [6] as instruments defined the file *clm-ins.scm.* [2]

The cross-fade solution has not been incorporated in common FM vocal synthesis implementations. Today, the most notable is the **FMVoice** instrument included in the Synthesis Tool Kit (STK) [7]. The class *FMvoices.cpp* can be freely downloaded as part of STK's source code and has been ported to various platforms e.g., Chuck [8] and Max / MSP / PeRColate [9]. In porting this class to Faust [10] and dealing with the discontinuity problem, I subsequently "rediscovered" for myself Le Brun's early solution. The same cross-fade solution also appears in Lazzarini and Timoney [2].

3. NEW PROBLEM

Lazzarini and Timoney also describe a method for generating formants with phase-syncronous oscillators. Its importance will become apparent. After adopting Le Brun's code in my own work and verifying that the discontinuity's clicks were gone (Fig. 2) I noticed that the fix introduced a new problem. This was again an audible artifact plaguing vibrato. Not clicks, but a new kind of artifact. Where perfectly periodic vibrato should elicit perfectly periodic spectral modulation it in fact, didn't. From one vibrato cycle to the next an overlayed pattern of spectral modulation is heard. The problem arises from phase mismatches in the pair of formants (even and odd harmonic numbers) being mixed for each formant. These are the pair being cross-faded to combat the clicks in what I will now label as the "first-order problem."

The cross-fade technique assumes that the energy of all coincident pairs of spectral lines will sum arithmetically. However, this assumption does not take phase into account. A "second-order problem" is caused by phase interference of coincident spectral lines. These are the spectral lines (carrier and sideband frequencies) of the two overlapping formant generators which fill out the spectral envelope of the formant. They are identical spectra which are shifted relative to one another by one harmonic number. All phases are generated relative to their respective carrier oscillators rather than to the ensemble of frequencies as a whole. And without phase-synchronous oscillators, these phases are arbitrary in time since they are independently determined by control changes.

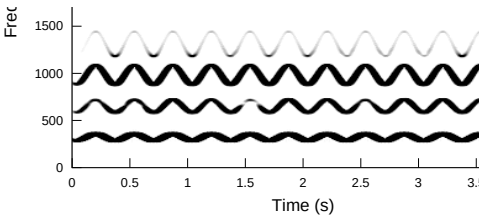
As discussed in Sec. 2 the first-order artifact is only apparent under changing conditions of pitch and phoneme target. Similarly, the second-order effect may remain unnoticed under steady-state conditions. With no change to carrier frequencies, carrier phases will also be constant and so will the resultant spectral mix. Nevertheless, interference between unrelated sets of phases can have an effect which alters the static spectral envelope and is perceived as a quality mismatch away from a target steady-state vowel. The problem becomes more apparent when carrier frequencies are being shifted dynamically, especially if these changes are happening periodically. Vibrato is a good way to emphasize the problem. Spectral distortions which may be imperceptible under other conditions are easier to hear with control changes which are repeating. The ear can pick out the distortion effect as a kind of spectral "isorhythm" or aliased pattern which is superimposed. Vibrato with a given period will generate a longer-period pattern of spectral modulation as seen in Fig.3. If you study the regions around 700 Hz and 1200 Hz, you will notice patterns in which phase-related Moiré fringing is inscribed on the amplitudes of the harmonics.

3.1 Minimizing Fringing

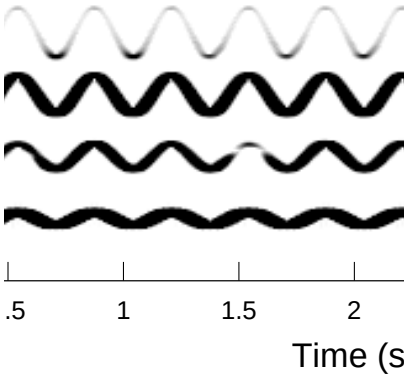
An initial attempt to minimize the audible effect of phase fringing is worth mentioning even though it isn't ultimately the solution being adopted. It exploits the fact that phase interference is most notable when the cross-fade mix of the two carriers approaches equal portions (when interaction will be greatest). This is the point at which the carriers are equidistant from the target center frequency. Conversely, the least interference occurs when one of them is closest to the target and the other is essentially muted. Taking advantage of this proportion where one oscillator dominates, by expanding its time in the (vibrato-related) duty cycle, is one way to minimize fringing.

In listening tests, it was found that the cross-fade ramp can be made non-linear and still mask the first-order discontinuity perfectly. By using a power law for the ramp slope, fringing is reduced by causing less time to be spent in the portion of the duty cycle with the problematic mix ratio. Initial experiments under periodic vibrato condi-

[2] One caution: some implementation versions belonging to this family have mistakenly labeled carrier oscillators as "modulators" and the reverse: their "carrier" is actually the modulator.



(a) fringing pattern changing over several vibrato periods



(b) fringing detail

Figure 3. Phase-related Moiré fringing, zoomed in from Fig. 2(a). When either of the two cross-faded carriers resets its frequency because the harmonic ratio needs to shift, it also resets its phase with respect to the other carrier. The result is a Moiré fringing effect visible in spectrograms.

tions indicated that even a very significant exponent can be used e.g., $f(x) = x^7$ and still mute the first-order artifact smoothly enough to avoid a click. This greatly reduces the time spent in "phase interference mode" and all but eliminates the audible effect of the second-order problem.

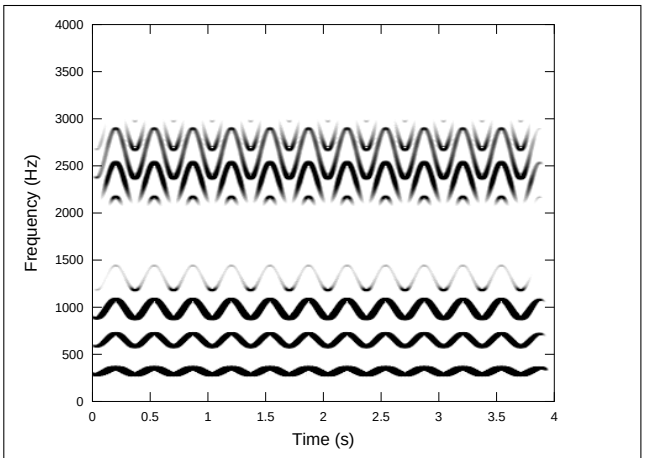
This fix has its drawbacks. Some fringing still remains during the portion of the duty cycle where the cross-fade mix briefly crosses through the equal portion region. More significantly though, is that altering the temporal dynamics of the mix distorts the mix away from the target i.e., away from the mix which best approximates the intended formant center frequency.

3.2 A Better Way

The root of the problem is in the use of independent oscillators. The way around this is to employ a bank of linked oscillators such as the phase-synchronous oscillators described in [2]. Here, a single phasor is shared by the modulator and all carriers. In the present implementation the bank can be constructed with any number of harmonic outputs and all will be tapped off of a single common phasor.

The familiar sampled sinusoid generates the fundamental (pitch) frequency signal for the bank to be constructed:

$$x(t) = Asin(\omega t + \phi) \quad (1)$$



fringing free

Figure 4. Re-rendering the vibrato example from 2(a) with phase-synchronous oscillators eliminates the fringing artifact.

where A is oscillator amplitude and ω in rad/s is calculated as $2\pi f$, where f is frequency.

Expressed in pseudo-code, Eq. 1 can be implemented with the modulo function:

```
w = f / SR
mp = 0.0
for n = 0 to N
  y[n] = a * sin(2pi * mp)
  mp = (mp + w) mod 1.0
end
```

The constant SR is the sample rate and the variable mp is the fundamental's instantaneous phase.

The key to the next step is sharing the instantaneous phase mp with any other oscillators, where o specifies oscillator number, cp_o its instantaneous phase and h_o its harmonic number:

```
cp[o] =  (h[o] * mp) mod 1.0
y[o][n] = a[o] * sin(2pi * cp[o])
```

and since we're interested in doing FM

```
m[o] = y[n] * i[o]
cp[o] =  (h[o] * mp) mod 1.0
y[o][n] = a[o] * sin(2pi * cp[o] +
                m[o])
```

The above pseudo-code implements one simple FM pair consisting of an independent carrier and shared modulator which produces a formant centered at harmonic h of pitch frequency f with modulation index i. The latter coefficient determines formant bandwidth and is typically used in a low range (< 2.0). In practice, a bank of six (or more) carrier oscillators of this kind will be used to generate a vocal sound. These will create phonemes of 3 (or more) formants represented by a time-varying distribution of h, a, and i coefficients.

The completed glitch-free method consists of Chowning FM singing voice + Le Brun cross-fade algorithm (from Sec. 2) + Lazzarini phase-synchronous oscillator bank (from Sec. 3). Fig.4 displays a spectrogram of vibrato rendered using the fully-realized solution (Faust and Chuck code to generate the example are included in the program appendices following). Classic phoneme table synthesis using Chowning's method can now be extended to arbitrary dynamic behavior.

4. APPLICATIONS

The singing voice technique has been used in three projects. Sound examples for each of these can be found online [11].

4.1 Converting eCog Signals to Music

Electrocorticography (eCog) registers brain electrical activity directly from inside the skull. Using sensors placed in regions suspected of giving rise to epilepsy, eCog arrays provide precise diagnostic monitoring as well as signals of great importance for studying the brain itself. In a current sonification project, the data is sung by a digital chorus. Each singer is a vocal simulation synthesized by the present technique. Where arrays have been implanted for therapeutic reasons, a large number of sensors is available (> 50) and the chorus can be made equally large. The aim of the work is to create a music directly from these sensors.

A correspondence exists between the temporal structures of music and the dynamics of brain activity monitored by eCog. Musical time has its notes, rhythms, phrases and epochal structures. Brain signals are marked by structures on the same time scales. Translation to music requires no modification of time base. In fact, the present approach avoids "re-composing" or altering the data in any way. The individuals who have contributed data to this project share our interest in discovering the potential of music as a different, new way of comprehending the complexity of brain dynamics.

Seizures we have listened to have a characteristic progression. They arise with a light, fast modulation "aura" not unlike a super-fast vibrato or tremolo which gives way to a nearly regular strong march of pulses. Multiple trains of pulses play against each other polyrhythmically. This crescendoes to an almost unbearable climax, the apex of the seizure. When it seems impossible for it to grow further, there is an abrupt cessation revealing a state of nothing. The paroxysm has switched off and the music is a quiet, calm, sustained chord. Motion is regained after this repose, but it is in a new world. Long, slow, undulations characterize the postictal phase whose affect is troubling, almost nauseous. Typically, this can last 45 minutes until normal brain activity is regained.

The method for translating brain signals into music is to have them modulate synthesized tones. The choice of singing synthesis makes an aesthetic connection to the "human-ness" of the data. Our chorus of eCog channels "performs" via modulations of pitch, loudness, vocal qualities and spatial location.

4.2 Speech synthesis

Speech synthesis, with its widely varying pitch and phoneme transitions, provides a good "real-life" test of the formant synthesis technique. The test has been created with a "toy" analysis – resynthesis platform driving synthesis from digitized singing and speech. The formant tracking analyzer is written in Chuck and the formant synthesizer is a Chuck UGen (unit generator) written in Faust. The analysis portion is FFT-based and uses a relatively long (4096 sample) window for formant accuracy (at 48 kHz sample rate). An example speech input fragment and the method's resynthesized output are compared in the spectrograms of Fig.5. Signal coding in this version consists simply of recording formant parameter updates which are relatively sparse (and could be greatly optimized). The results are promising for developing this into an FM-based speech coder – the example consists of two different speakers in a heated, emotional dialog. Their voices and identities are preserved, as is their expressive prosody and intelligibility. The analysis tracks populations of short-lived formants which in the example are limited to 4 at a time (using 9 oscillators total).

4.3 *Near the Inner Ear*

The formant generation technique was applied in a recent composition for orchestra, computer music and video premiered in 2013 by the Stanford Symphony Orchestra, Jindon Cai, conductor. The orchestral score by Dohi Moon was recorded and analyzed with a formant tracking algorithm. Analyzed formant tracks were also obtained from a recording of the first movement of Beethoven's Ninth Symphony (the two pieces were performed together for the conclusion a full Beethoven cycle). *Near the Inner Ear* incorporated resynthesized clips from both analysis data sets.

The composition used formant-based resynthesis to generate a novel timbral identity whose music is tied to the orchestral writing but whose comportment is different enough to constitute a kind of musical "alter ego." The work opens with a 90 s section in which an antiphonal exchange exploits such contrasts. The resynthesis instrument returns at various moments during the piece with musical gestures which reinforce the accompanying video composition by John Scott.

5. CONCLUSION

One goal of the analysis – resynthesis system going forward is to create a large database of acquired vocal sounds in order to structure more complex phoneme-based singing synthesis from a vastly expanded table. Applications described above have demonstrated a potential to create rich reservoirs of articulations and timbral identities. The hope is tap into greater timbral variety for the sonification work. Possibly also to acquire listeners' own voice traits for EEG-driven synthesis.

Resynthesis in the service of music projects like *Near the Inner Ear* can also take liberties with acoustic structures in order to create new instrumental identities. In *Near the Inner Ear*, the formant tracker operating on orchestral sound

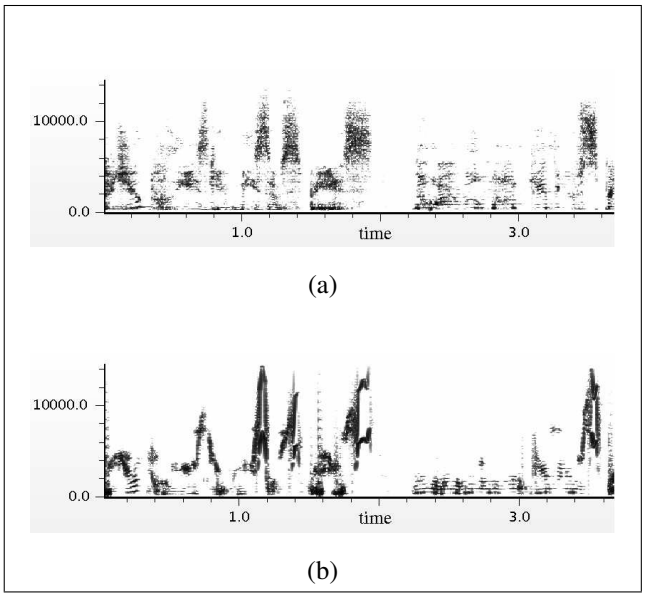


Figure 5. Analysis – resynthesis of dialog: (a) is the input from an argument between a teenage daughter and her mother, "can't you please give me some space," and "no, I will not give you some space.", (b) FM resynthesis

inferred formants where no model would predict them to exist. Experimenting with resynthesis, it was found that often the tracker would emphasize pitches of inner voices in the analyzed recordings. Rather than impose an $f0$ fundamental pitch, formant frequencies themselves were allowed to become the $f0$. The remaining formants would then create a kind of "instrumental singing" whose resonances mapped to the pitch structures from the original recordings.

The present synthesis method can be used for non-vocal sounds whose acoustic structures are also represented with formant-like resonances. Horner has explored timbre matching for a sampled trumpet using a genetic algorithm to find suitable FM formant parameters [12].

The improvements to FM vocal synthesis detailed in this paper can be extended to other audio rate modulation schemes, in particular those which also employ single modulator / multiple carrier structures. A glitch-free AM vocal synthesis "cousin" has also been implemented in Faust. AM has the advantage of simplicity in prediction of dynamic sideband behavior (AM sidebands are free of the Bessel function which determines FM sidebands).

Acknowledgments

Many thanks to John Chowning for his inventions and encouragement, musical and technical. Bill Schottstaedt continues to passage into the future comprehensive sets of synthesis instruments and analysis tools. His Snd project preserves and provides essential computer music algorithms without which much of the present work would not have been possible.

6. REFERENCES

[1] J. Kleimola, "Nonlinear abstract sound synthesis algorithms," Ph.D. dissertation, Aalto University, Helsinki, Finland, 2013.

[2] V. Lazzarini and J. Timoney, "Theory and practice of modified frequency modulation synthesis," *J. of Audio Eng. Soc.*, vol. 58, no. 6, pp. 459–471, 2010.

[3] J. Chowning, "Computer synthesis of the singing voice," in *Sound Generation in Winds, Strings, Computers*, J. Sundberg, Ed. Royal Swedish Academy of Music, 1980, pp. 4–13.

[4] ——, "Frequency modulation synthesis of the singing voice," in *Current Directions in Computer Music Research*, M. Mathews and J. Pierce, Eds. MIT Press, 1989, pp. 57–64.

[5] M. L. Brun, "Digital waveshaping synthesis," *J. of Audio Eng. Soc.*, vol. 27, no. 4, pp. 250–266, 1979.

[6] "Scheme, Ruby, and Forth Functions included with Snd," last viewed 29 Mar. 2013. [Online]. Available: https://ccrma.stanford.edu/software/snd/snd/sndscm.html

[7] "FMVoices Class Reference, in The Synthesis ToolKit in C++," last viewed 29 Mar. 2013. [Online]. Available: https://ccrma.stanford.edu/software/stk/

[8] "Chuck : Strongly-timed, concurrent, and on-the-fly audio programming language," last viewed 29 Mar. 2013. [Online]. Available: http://chuck.cs.princeton.edu/

[9] "PeRColate, A collection of synthesis, signal processing, and image processing objects for Max/MSP," last viewed 29 Mar. 2013. [Online]. Available: http://music.columbia.edu/percolate/

[10] "FAUST (Functional Audio Stream)," last viewed 29 Mar. 2013. [Online]. Available: http://faust.grame.fr/

[11] "Supporting online materials." [Online]. Available: http://ccrma.stanford.edu/~cc/vox/smac2013som/

[12] J. B. A. Horner and L. Haken, "Machine Tongues XVI: Genetic algorithms and their application to FM matching synthesis," *Computer Music J.*, vol. 17, no. 4, pp. 17–29, 1993.

7. PROGRAM A

The Faust program, FMVox.dsp [11], implements an FM-voice algorithm with four formants consisting of uniform phase table-lookup harmonic oscillators. A multiplexed audio rate coefficient stream drives the synthesis. For each formant a demuxer is included which extracts its coefficients from the stream. Formants are instantiated in a parallel composition by the Faust process which outputs the sum of their signals. The resulting unit generator compiled by Faust can have a scalable number of formants. A texture

of multiple voices can be created from multiple, independent voice control streams which flow to independent unit generators. Using this architecture a choir of voices can be distributed across multiple sample-synchronous threads and / or multiple cores.

```
declare name "FMVox";
import("filter.lib");
ts  = 1 << 16;
fs  = float(ts);
ts1 = ts+1;
ct  = (+(1)~_ ) - 1;
fct = float(ct);
sc  = fct*(2*PI)/fs:sin;
sm = fct*(2*PI)/fs:sin:/(2*PI);
dec(x) = x-floor(x);
pha(f) = f/float(SR):(+:dec) ~ _;
tbl(t,p)= s1+dec(f)*(s2-s1)
with {
  f = p*fs;
  i = int(f);
  s1 = rdtable(ts1,t,i);
  s2 = rdtable(ts1,t,i+1); };

fupho(f0,a,b,c) = (even+odd):*(a)
with {
  cf  = c/f0;
  ci = floor(cf);
  ci1 = ci+1;
  isEven= if((fmod(ci,2)<1),1,0);
  ef  = if(isEven,ci,ci1);
  of  = if(isEven,ci1,ci);
  frac = cf-ci;
  comp = 1-frac;
  oa  = if(isEven,frac,comp);
  ea  = if(isEven,comp,frac);
  ph  = pha(f0);
  m  = tbl(sm,ph):*(b);
  even = ea:*(tbl(sc,(dec(ef*ph))+m));
  odd = oa:*(tbl(sc,(dec(of*ph))+m));};
frame(c) = (w ~ _ )
with {
  rst(y)= if(c,-y,1);
  w(x)  = x+rst(x); };
demux(i,ctr,x) = coef
with {
  trig = (ctr==i);
  coef  = (*(1-trig)+x*trig) ~ _;};
formant(f_num,ctlStream) = fsig
with {
  ctr = frame(ctlStream<0);
  co(i) = demux(i,ctr,ctlStream);
  f0  = 1;
  a  = f0+1+f_num*3;
  b  = a+1;
  c  = a+2;
 fsig  = fupho(co(f0), co(a),
     co(b), co(c)); };
nf = 4;
process = _<:par(i,nf,formant(i)):>_;
```

The **fupho** function implements one formant (of uniform phase harmonic oscillators) which is free of glitches caused by dynamic behavior. Its inputs are the fundamental frequency, formant amplitude, bandwidth and center frequency (respectively, a, b, c). A **fupho** receives demuxed controls via its calling function **formant**.

8. PROGRAM B

FMVox is used in the Chuck program, FMVoxVib.dsp [11], to produce Fig.4. The example defines a master "shred" which executes for 4 s. It sets up a DSP graph in which one **FMVox** instance receives a sample-synchronous control stream and sends its output to the dac. The master shred "sporks" child shreds for vibrato and a multiplex control stream. The **data** float array holds "aaa" vowel coefficients for four formants described by their amplitude and center frequency. For this test code formant bandwidths are set the same globally.

```
Step stream => FMVox fmv => dac;
4 => int nFormants;
1::ms => dur updateRate;
SinOsc vibLFO => blackhole;
vibLFO.freq(3);
vibLFO.gain(0.1);
Std.mtof(64) => float p => float f0;
fun void vibrato() {while (true){
  ((vibLFO.last()+1.0)*p) => f0;
  1::ms => now;
}}

[ // "aaa"
  [ 349.0,  0.0],[ 918.0,-10.0],
  [2350.0,-17.0],[2731.0,-23.0]
] @=> float data[][];

fun void mux(float val) {
  stream.next(val);
  1::samp => now;
}

-1 => int startFrame;
95 => float db;
fun void muxStream() {
  updateRate-14::samp => dur padTime;
  while(true){
    padTime => now;
    mux(startFrame);
    mux(f0);
    for (0 => int f; f<nFormants; f++){
      mux(Math.dbtorms(db+data[f][1]));
      mux(0.2);
      mux(data[f][0]);
    }
}}

spork ~muxStream();
spork ~vibrato();
4::second => now;
```