Real-Time Digital FM Audio Synthesis

by Steve Saunders

Carnegie-Mellon University

Xerox Palo Alto Research Center

ABSTRACT:

Chowning's method of synthesizing complex audio tones, for computer-generated music, by frequency modulation produces a remarkable range of useful, interesting, controllable sounds given a very few parameters. However, this method requires too much arithmetic to be implementable for real-time generation of several voices on available minicomputers. I describe a form of FM synthesis, using a triangle-wave modulating signal, which exhibits the same virtues of versatility and controllability but which is implemented for real-time music on a modern minicomputer with no music-specialized hardware but a DAC.

1. Motivation

"The computer as a musical instrument" has been a popular theme and goal for some years now - and rightly so, for the versatility-under-control of the digital computer is unsurpassed by that of any other man-made device. From the early days of singing line-printers and square-waves to MUSIC V [Mathews] and more recent systems, great effort has gone into realizing the musical potential of our mathemtical engines.

Only recently, however, has it become feasible to consider the computer as a performing instrument, with immediate feedback to the performing musician (along with recording, editing, and mixing capabilities, of course). Recent minicomputers are just sufficiently powerful to do the necessary interaction and computation, and just cheap enough to be competitive with more traditional methods of sound generation.

2. Chowning FM

In 1972, John Chowning at Stanford A. I. Lab invented a new method of synthesizing musical sounds that is both extremely simple and exceedingly versatile [Chowning].

Chowning's method consists of frequency modulation of a sine-wave carrier (at the desired fundamental frequency) by a sine-wave signal at the same or a closely related frequency. The resulting spectrum contains components whose amplitudes are given by Bessel functions of the first kind, $J_n(X)$, where n denotes the nth harmonic, and X is the modulation index (the ratio of the peak frequency deviation to the modulating signal frequency: a dimensionless parameter).

Considering the case where the carrier and signal have the same frequency, the upper side frequencies lie in the harmonic sequence, as one would expect; but the lower sideband "reflects" about zero (DC) and

overlaps the same harmonic sequence. So a musical (harmonic) spectrum is indeed produced.

Now what is the effect of the parameter X, the modulation index? Consulting any standard book of tables we see that for X = 0, only $J_0(X)$ is nonzero (in other words, only the fundamental occurs). As X increases, the first few harmonics become non-negligible, and the fundamental decreases. That is, the spectrum "spreads out", acquiring contributions from ever-higher harmonics of the fundamental. Notice that when X > 1, the instantaneous frequency is sometimes negative. This poses no difficulty for digital generation, though.

So we see that by varying one parameter, modulation index, we can obtain an entire family of tone colors - from a pure sine wave to a "nasal" or "brassy" spectrally-topheavy sound.

But that's not by any means all! Any of these spectra can be produced by conventional methods - just compute and store one cycle, and play samples from it at a pitch-determined rate, for example. But these sampling methods always sound "electronic" or lifeless, because the spectrum they produce almost never varies much. With FM, on the other hand, we can get tones which evolve in time - the way a trombone goes "Blaaaatt", or a big bell gradually settles down to a dim, pure tone. This spectral evolution through time apparently provides much of the "signature" information by which we recognize various instruments and judge timbre; it is surely an important ability for any device trying to be a universal muscial instrument.

3. More Details

The process of generating an FM tone digitally can be better understood from the following viewpoint: the instantaneous frequency is just the rate-of-change of phase. Phase is represented by an index into a sine-wave lookup table or argument to a sine function; the rate-of-change is just the increment added to this index between samples. If we now recompute the index step (= instantaneous frequency) for each sample, we have the capability for FM.

In Chowning's work, the modulating signal is a sine wave. This implies that the increment computation is

$$freq(t) = f_0 + f_m X \cos(f_m t),$$

where f_0 is the carrier frequency, f_m is the signal frequency, X is the

modulation index, and t increases linearly with time.

The sinusoid character of both carrier and signal allows the analysis of the spectrum in terms of Bessel functions given by [Chowning] and [Van der Pol]:

$$\sin (f_0 t + X \sin (f_m t)) =$$
$$J_0(X) \cos (f_0)t$$
$$- J_1(X) \cos (f_0 \pm f_m)t$$
$$+ J_2(X) \cos (f_0 \pm 2f_m)t$$
$$- \ldots + \ldots$$

4. Triangle FM

However, it appears that the exact composition of the spectrum is not nearly as important as the overall character and the time-evolution of the tone. Examples abound of uncannily real "trumpet" and "trombone" sounds produced via FM - the wave shapes and detailed spectra are not at all trumpetlike, only the sensation is.

If we are willing to give up the simple mathematical analysis of our spectra, while retaining the single-parameter control over its extent and general character, a much easier form of FM presents itself. Note in the sine-by-sine modulation step that at least one multiplication seems to be required to implement the effect of the modulation index on the signal. Most microprogrammable minicomputers do not execute multiplications at the requisite rate for real-time generation of several voices with time out for control (i.e. reading the input keyboard to see which notes to play!).

Suppose, now, that we use for the modulating signal a triangle wave instead of a sine. The increment computation becomes

$$\text{freq}(t) = f_0 + f_m \times \text{tri}(f_m t).$$

This can be implemented by starting freq = f_0, and then just doing

$$\text{freq}(t) = \text{freq}(t-1) + I,$$

where I is the constant $f_m^2 \times L$.

(L is an implementation factor which accounts for the length, in samples, of the sine-wave table.)

We also require a comparison to insure that freq is between $f_0 - f_m \times$

and $f_0 + f_m X$. When freq gets outside its limits, we merely "turn the triangle around" by changing

$I = -I$.

Note that the freq (t) computation now involves only adding one pre-computable constant!

A more exact specification of my FM-synthesis method, in approximate ALGOL, follows (this is the microcoded loop for one voice):

```
procedure FM (f0,fm,X,numsamples);
  integer numsamples; real f0, fm, X;
  begin
    real increment, freq, phase, upperlimit, lowerlimit;
    integer T;

    increment := fm*fm*X*lengthfactor;
    freq := f0;
    upperlimit := f0 + fm*X;
    lowerlimit := f0 - fm*X;
    phase := 0;

    for T:= 1 step 1 until numsamples do
      begin
        freq := freq + increment;
        if increment > 0 then
          If freq > upperlimit then
            begin
              increment := - increment;
              freq := 2*upperlimit - freq;
              comment This just "reflects" freq about the
                limit, getting it as far inside as it was
                outside;
            end
          fi
        else
          if freq < lowerlimit then
            begin
              increment := - increment;
              freq := 2*lowerlimit - freq
            end
          fi
        fi;
        phase := (phase+freq) mod cyclelength;
        enqueue (sine [entier(phase)])
      end
  end FM;
```

This procedure assumes an array "sine" of "cyclelength" DAC-acceptable samples which will yield a sine wave if stepped through sequentially, and a procedure "enqueue (sample)" which puts the given sample into the buffer for the DAC process. Of course, the microcoded version uses carefully scaled integers instead of floating-point reals.

The exact analysis of the spectrum produced by triangle-wave modulation is rather more difficult than the sine-wave case. For the triangle-wave modulation, following [Crosby], I get

$$
\begin{aligned}
& J_0(x_1)\, J_0(x_3)\, J_0(x_5) \ldots \sin (f_0)t \\
& - J_1(x_1)\, J_0(x_3)\, J_0(x_5) \ldots \cos (f_0 \pm f_m)t \\
& - J_2(x_1)\, J_0(x_3)\, J_0(x_5) \ldots \sin (f_0 \pm 2f_m)t \\
& + [J_3(x_1)\, J_0(x_3)\, J_0(x_5) \ldots \\
& \quad + J_0(x_1)\, J_1(x_3)\, J_0(x_5) \ldots] \sin (f_0 \pm 3f_m)t \\
& + [J_4(x_1)\, J_0(x_3)\, J_0(x_5) \ldots \\
& \quad + J_1(x_1)\, J_1(x_3)\, J_0(x_5) \ldots] \sin (f_0 \pm 4f_m)t \\
& - \ldots
\end{aligned}
$$

where x_n is the modulation index corresponding to the nth harmonic of the triangle wave,

$$x_n = 8/\pi^2\, X/n^3 \,, \quad n = 1, 3, 5, 7, \ldots$$

The x_n are derived from the expansion

$$
\begin{aligned}
\text{tri}\,(f_m t) = 8/\pi^2 \,[&\sin f_m t - 1/9 \sin 3f_m t + \ldots \\
&\pm 1/n^2 \sin n f_m t \ldots] \,;
\end{aligned}
$$

the extra factor of 1/n comes from the fact that the frequency being considered is $n f_m$.

Noting that $J_0(0) = 1$, $J_n(0) = 0$ for $n > 0$, and that $J_n(x)$ becomes significant about when $x = n-1$, we see that the triangle-FM spectrum behaves as required: with zero modulation index, a pure sine wave results; with nonzero X, the spectrum is harmonic (it contains components at each sum and difference of the carrier and multiples of components of the signal); as X increases, the spectrum must generally spread out, with more energy going into higher harmonics; and the individual spectral components change and grow in complex ways.

Crosby remarks that FM with multi-tone or "program" signal produces smaller out-of-band components than pure-sine modulation with the same over-all modulation index; he refers to band-limited signals, however. In our experience, using triangle waves instead of sines has a similar effect to using a 1.5 to 2 times larger modulation index with sine signal.

5. Advantages

The advantage of triangle-signal FM synthesis over sine-signal FM is

simply speed resulting from computational simplicity. Of course, fast multiplication hardware would eradicate this edge completely. Its advantages over other methods of musical-sound synthesis for computers are precisely those of sine-sine FM as described by Chowning and mentioned above.

6. Volume Control

It may be objected at this point that all this FM trickery would still make quite dull sounds because it does not incorporate dynamic (loudness) variations as well as spectral modulation, especially in the critical attack portion of a note's lifetime; and that this seems to require a multiplication per sample after all.

The solution, suggested by Steve Purcell, is to note that

$$\sin (x + d) + \sin (x - d) = 2 \cos d \sin x .$$

So if, instead of fetching just one sine sample at the end of the procedure given above, we fetch two, displaced from the computed phase by an amount which is a parameter of the procedure, we can have any amplitude of FM waveform we like, at a net cost of two additions for addressing and one more memory cycle! The required phase-displacement parameter is given by

$$d = (\text{cyclelength} / \pi) \operatorname{Arccos} (\text{ampl} / 2)$$

for an amplitude of ampl relative to the stored sine wave. The necessary modifications to the above program should be clear.

7. Another FM Method

We can, it seems, apply this same trick to computing the modulating signal for the FM computation, eliminating those hard-to-analyze triangle waves:

```
procedure NewFM (f0,fm,X,numsamples);
  integer num;  real f0, fm, X;
  begin
    real phase, time;
    integer T, Xoffset;

    Xoffset := entier(lengthfactor * Arccos(fm*X/scale/2));
    phase := 0;

    for T := 1 step 1 until numsamples do
      begin
        time := time + fm;
        phase := phase + f0
                    + sine2[(time+Xoffset) mod sine2length]
                    + sine2[(time-Xoffset) mod sine2length];
        enque (sine[entier (phase)]);
      end
  end NewFM;
```

This assumes, besides what the triangle case needed, a second array "sine2" containing "real" values scaled up (by a factor "scale") to the maximum modulation index - frequency product that we choose to allow. This implementation requires, however, great dynamic range in the amplitude control of the modulating signal, and therefore a very large sine table.

8. Implementation

On current minicomputers, the only apparent way to get the necessary speed of FM synthesis (10,000 samples/sec or more), even with this simplified approach, is through use of microcode to do the innermost loop. Also we can gain speed by keeping the intermediate stages in high-speed registers through several samples of the wave. We can best do this by making a single entry to the microcode ("macro-instruction") cause generation of many samples.

The computer cannot spend all its time in this inner loop, however, since it must respond to control inputs from the performer (e.g. key pressed/released, stop pulled in the case of an organ console) within 10-20 ms.

One solution which has been tried with some success is to run the control and setup code on a 60Hz-interrupt basis. This code checks for changed control inputs, sets up the necessary parameters for the FM microcode, calls that microcode, and quits. The FM microcode, when called, produces enough samples to fill 1/60 second, places them in a buffer queue, and returns.

A separate parallel process (e.g. a DMA device) reads samples from the queue to a D-A converter at a constant sample rate.

This separation has the advantage that low-priority processing, changes of global state, TTY I/O, debuggers, etc. can make free use of any time not required by music generation - typically a few percent of the machine. Such "background" actions might include turning music on or off (by disabling the 60Hz interrupt, for example), reassigning meanings to the control inputs, reading in a block of "compiled" or "recorded" control information to play a specified tune, or running a general-purpose language interpreter in which any or all of the above can be accomplished by typed commands. The separation into parallel, separately clocked processes allows music generation to proceed undisturbed by other, possibly totally unrelated, processing.

9. Experimental Results and Plans

The first method I described has been programmed on an inexpensive minicomputer and is incorporated in a full interactive music system, with facilities for playing on an organ keyboard, recording, editing, overdubbing, displaying, and printing music. The performance of our system can be summarized as follows:

a) 5 real-time FM voices
b) 12-bit samples at 13.7 kHz rate
c) 60 Hz updating of all FM parameters: carrier and signal frequencies, modulation index, and volume
d) Arbitrary parameter envelopes with an attack transient, a repeated sustain, and a decay section
e) Very conventional, cheap, general-purpose, microprogrammed minicomputer
f) No music-specific hardware but the DAC

We intend to explore methods to facilitate searching the space of timbres available through FM, and other methods of input and control (drumheads, microphones, ADCs ("hum a few bars"), etc.).

REFERENCES:

J. M. Chowning

"The Synthesis of Complex Audio Spectra by Means of Frequency Modulation"

J. Audio Eng. Soc. 21, pp 526-534, 1973

M. G. Crosby

"Carrier and Side-Frequency Relations With Multi-Tone Frequency or Phase Modulation"

RCA Review 3, pp 103-106, 1938

M. V. Mathews

Technology of Computer Music

M.I.T. Press, 1968

B. Van der Pol

"Frequency Modulation"

Proc. IRE 18, pp 1194-105, 1930