SY Programming

Herbert Janßen
heja@neuroinformatik.ruhr-uni-bochum.de

Version 0.60 (LaTeX2e), December 17, 2002

Contents

1 Copyright

OK, here we go:

This document is available as a postscript formatted and zip compressed file in `ftp.neuroinformatik.ruhr-uni-bochum.de:/pub/outgoing/heja/sy-list/MISC/sy-prog.zip` and on the world wide web as an online browsable hypertext document in HTML via `http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/heja/sy-prog/sy-prog.html` The HTML version will usually be less up to date though.

And of course the mandatory sentence: The contents of this document reflect my opinions only and not necessarily those of my employer or anyone else. And - by the way - I am not or have ever been affiliated with Yamaha.

2 Introduction

This text is a heterogeneous compilation of ideas about synthesis and SY programming that I wrote down over time, mostly in a scratch paper fashion. It is also a fragment, although it will hopefully increase in size and gain consistency in the future. It is essentially about the SY99 - since this is the only SY I really know - but nearly everything is applicable to the SY77 and TG77 and a few things even to other synths.

The general approach is to give anybody who is interested a collection of information about synthesis and hints and ideas on how to program the SY to use its full potential. My motivation for doing this is that I myself was desperately looking for any information on that matter for years now and have found only rudimentary sources, so I contribute another fragment.

All this is my personal view of things which is based on true diletantism, i.e. love of the subject of synthesis without any professional training or interest. Thus, I am very interested in any useful commentary on everything, especially omissions and errors, which I know must be quite plentiful.

Many of the thoughts contained in this text are actually based on e-mails and postings or inspired by discussions with various net inhabitants as well as by numerous articles and books. Thanks to all the people on the SY-List and other places on the internet for sharing ideas, information, opinions and enthusiasm. Let me name just a few people who definitely deserve credits: Adam Mirowski for starting and administering the SY-List, Ab Wilson for a lot of excellent thoughts about FM and synthesis, Ken Beesley for his very informative posts about wind-controlled synthesis, correcting errors in this document and other valuable help, Christian Stiens and Argiris Kranidiotis for various details on the FM implementation and tips for voice synthesis, Ulrich Köpping for collecting the "SY frequently asked questions and answers file" (SY-FAQ), Jeff Harrington for all the microtunings, Georg Müller for lots of miscellaneous help. And Martin Czech for setting me straight with respect to FM - PM differences.

And a very very special thanks to Michael Neef, our SysOp, who provided disk space, helped with network and software problems and always has a friendly attitude towards providing free information and software.

Synthesis isn't about creating sounds, it's about creating instruments, complete with their own expressive characteristics, and then playing these new instruments using the characteristics you've invented.
Ab Wilson

3 Synthesis Methods

A good background for trying to program a synthesizer is to know about the different synthesis methods developed over the years. I think the most important reason for knowing these is that they represent different paradigms of access to sound: many timbres can be created with almost any of the synthesis methods discussed, but each of them has its strong and weak points and requires its own way of "thinking" and playing - which may be useful to develop.

The most important difference between the synthesis approaches is - in my opinion - not so much in the kind of sounds they are able to produce, but in the programming and the available control parameters. Below I have tried to list up the main synthesis approaches and their general properties in relation to the following criteria:

- The ability to automatically re-create or resynthesize a given sound. Sampling is a special case here since it allows immediate playback with unsurpassed fidelity but does not offer a way to modulate the sampled timbre without special programming.
- The ease of programming, i.e. the predictability and power of the synthesis parameters in the hands of an experienced user.
- The amount of data that is needed to describe a certain patch. Big amounts of data limit handling and accessibility.
- The necessary hardware to implement a synthesis method.
- The interactive control possibilities which should be able to allow a musical relevant access to timbre via phrasing and special control parameters.
- The sound versatility and quality. A good measure for this is how "natural" an acoustical instrument emulation will sound when played on the synthesizer.

	additive	classical subtractive	sample playback	frequency modulation	physical modeling
resynthesis	+	−	(+)	∘	−
programming ease	−	+	+	−	−−
hardware complexity	∘	+	+	+	−
amount of data	−	+	−	+	∘
interactive control	∘	+	−	+	++
sound versatility	+	−	+	∘	+

The table shows that each method has specific advantages and drawbacks. For me this implies that hybrid respectively modular synthesis architectures are the only way to build a general purpose synthesizer.

The technical terms for synthesis methods are very much mixed up with marketing buzzwords. I will not really try to distinguish between the two categories, but rather discuss the matter in colloquial terms, trying to be as exact as possible. All synthesis methods I ever heard of are listed below. Necessarily I do not have hands on experience with all of them. I tried to write only about things I feel I have really understood though...

3.1 Additive Synthesis, Fourier Synthesis

Any sound - however complex it may be - can be described as a mixture of a number of sine wave components with different phases and amplitudes. These are the *partials* of a sound. The partials are also called *harmonics* if their frequencies are integer multiples of a fundamental frequency (in this case the waveform is periodic).

The method to generate a complex sound spectrum as the sum of (many) simple sine waves is called Fourier synthesis - after Jean Baptiste Joseph de Fourier who found its mathematical basis. The more general term additive synthesis can also be used if the waveforms added are not necessarily sine waves.

Ideally, a lot of sine oscillators are needed for Fourier synthesis. How many depends on the required range and brightness: a bright bass note - think of a slap bass - may need more than hundred, while a high pitched harmonic sound will probably need only a dozen.

For dynamical sounds and expressive play of an additive synth very many parameters are needed: ideally, each oscillator should have its own amplitude envelope, pitch envelope, velocity sensitivity and modulation routing.

Although it may sound like the hardware is the limiting factor, the usability is even more so. Most of the many parameters have only little influence on the sound and generally it is extremely hard to estimate how the spectrum of a desired sound looks like. Thus the simulation of acoustic instruments seems to be impossible without appropriate analysis hardware and software. So if you think you have mastered FM, try additive for a change.

If you are looking for additive synths, you will find that the market is very limited: There are only two machines that are in an affordable price range, the Kawai K5 (or rackmount K5m) and the Kurzweil K150FS. The Kawai is more limited in architecture (only harmonic partials) and sound but has a very nice programming interface, while the K150FS is more versatile but can only be programmed via some Apple II software. The Synergy is another early digital synth that does FM and additive synthesis but it is rather rare. Kawai has recently announced a new additive Synth, the K5000 that - on paper - looks like it will be much more powerful than any additive synth before.

Some synths offer to generate waveforms by "drawing" their shape or specifying the levels of their partials. While the latter is a typical additive technique, it would not make much sense to call those synths additive, since they offer no realtime control of the additive parameters and therefore playing those additive waveforms is in no way different than playing a sample.

Some people call any synthesizer that is able to layer sounds additive; some call FM a variant of additive Synthesis (I saw this in a book); it should be obvious why this is at least misleading.

3.2 Subtractive Synthesis

This is *the* classical method of synthesis used in most analog synths and in most sample playback synths and samplers.

Subtractive synthesis means that you take a sound (preferably a spectral rich one like a sawtooth or a square/pulse wave, or a sample of a grand piano) and route it through a modulatable filter and amplifier to change its timbre. This way you reduce or "subtract from" the level of some partials of the original spectrum; hence the term "subtractive". The terminology is a bit fuzzy for the real world, since almost any synth uses filters. The general usage of the term tends to refer to the classical "oscillator - filter - amplifier" trinity though.

What is nice about subtractive synthesis is that by selecting the oscillator waveform or sample, the basic timbre is rather well determined, and the usual filter and amplifier parameters allow you to effectively and predictably tweak it to make it brighter, duller, more percussive etc.

The main problem with subtractive synthesis is that its tools are rather bold: the oscillator waveforms or samples have a distinct character that is hard to overcome and the usual filters do not allow for very detailed transformations.

To push subtractive synthesis further a lot of extensions to the basic subtractive synth scheme have been developed: complex filters (e.g. on the newer Emu Synths and Samplers), waveshaping, ring modulation and many more.

3.3 Analog Synthesis

Analog Synthesis is not really a synthesis method, but rather a hardware classification: analog synths use analog instead of digital electronics to create their sounds. What most of them can do in terms of synthesis methods is quite simple subtractive synthesis. However, some more advanced machines and especially modular synths may use a great variety of synthesis methods including FM, waveshaping, vector synthesis and others.

Analog synths are considered "cool" nowadays, because of their supposedly "warm" and "fat" sound. So what is so special about analog synthesis then?

First, most analog synths have an extensive user interface with dedicated knobs and switches for every function. Thus allowing very intuitive access and instant gratification for sound tweaking. This is possible because typical analog synths offer a limited number of control parameters, but those parameters are highly effective.

Second, rather simple analog circuitry can perform the functions needed for subtractive synthesis rather well: the resulting sound will be artificial but with slight variations and instability. Thus, the sound quality is lively in a way similar to acoustic instruments. On the other hand most digital synths compromise the sound quality in order to achieve the high number of voices many buyers seem to be fond of today.

Modular analog synths have the additional advantage that there is no distinction between audio and control signals. Everything is just a control voltage, which results in a vast number of patching possibilities. These beasts are very rare and pricey nowadays, but are probably still the best way to learn about synthesizing sounds, and can be used as musical instruments of exceptional power.

A little note: there are some digital instruments that contain what can be pretty good described as a digital implementation of the classical analog synthesizer architecture. I would count in the Roland D-50 and the Clavia NordLead here.

3.4 Sample Playback (PCM, AWM, AWM2, AI, ...)

Sample playback usually means a form of subtractive synthesis that is also called PCM (Pulse Code Modulation), AWM (Advanced Wave Memory), AWM2 (Advanced Wave Memory Version 2) AI (Advanced Integrated) and many other names by manufacturers. Usually all those terms refer to basically the same thing: An audio signal, e.g. a miked acoustic instrument or an electrical or electronical instrument, is sampled (digitized), and the recording is stored in RAM or ROM. If a device is able to sample and store the result in RAM or to disk, it is called a sampler. A device that can play back samples (from RAM, ROM or disk) at different pitches is called a sample playback synth. Most samplers and sample-based synths use subtractive synthesis although there are some samplers that offer only very limited processing (no dynamical filters) so that they cannot be considered synthesizers at all.

The term PCM refers to the coding technique which is used in virtually all digital instruments. The terms AWM and AWM2 are Yamaha marketing slang for 16-bit sample playback and 16-bit sample playback with filters. Korg uses the term AI synthesis for their M1, which is just another sample playback synth - synthesis wise.

Sample playback is what has made synths realistic sounding. On the other hand sampling per se offers less options for expressive play than almost any other synthesis scheme. Samples that have a lot of inherent character or are easily recognized as an acoustic instrument are hard to shape, so filters and other processing options will merely adjust the timbre of the sample.

There are however unique possibilities in sample synthesis, but these are often not implemented in commercial synths. I'm talking about modulation of the sample playback parameters themselves: extreme transposition of multisamples, modulation of sample start and sample loop length, multiple sample loops and more. Among others, various Ensoniq and Emu synths and samplers are capable of some of these. On many synths (including the SYs) even transposition (the changing of sample playback rate) can only be achieved with the trick of a constantly biased pitch envelope.

3.5 Frequency Modulation (FM, AFM)

Frequency Modulation is usually abbreviated FM or AFM (for Advanced Frequency Modulation). This is the family of synthesis methods that brought a breakthrough for commercial digital instruments in the eighties. Basically it means that you control the frequency of an audio oscillator by the frequency of another audio oscillator. The interesting aspect sound-wise is that you can generate a very wide variety of spectra plus many transient sound characteristics with FM (and not only the never ending variations of electric pianos and bells).

FM as a synthesis method was "invented" by John M. Chowning at Stanford and used in the academic computer music scene long before Yamaha marketed it. The commercial Yamaha implementation introduced some restrictions, but also some useful extensions like feedback.

FM exists in many different flavours: some analog synths respectively digital/analog hybrids are able to do a very basic FM. But FM becomes a versatile synthesis technique only with multiple oscillators and multiple envelopes to control their amplitude, which results in a big number of components/modules needed in the analog realm. Maybe this is why it was and is not as popular with analog synths. Also, the details of analog FM are quite different from Yamahas digital FM implementation (no negative frequencies, no phase parameter, frequency dependent feedback delay time).

Yamaha's digital FM implementations use custom chips to reduce cost. In the case of

digital FM, there are also many variants: depending on the number of oscillators (minimum is 2, most synths use 4 or 6, I recall having heard of 10 in some Yamaha organs), whether there is a real envelope per oscillator (some very simple Yamaha soundchips like the one used in the old Atari STs lack them) and of course how variable the routing between the oscillators is (number of "algorithms", modulation and feedback paths).

There are basically 3 types of FM that were implemented in Yamaha synths: the DX7 type 6-operator FM, some 4-operator FM variants, and the extended form of 6-operator FM in the SY77, TG77 and SY99. Yamaha calls the synthesis method of these newer 6-OP FM synths AFM. AFM is a *big* step in sonic capabilities from the former FM implementations: particularly the 3 programmable feedback loops enable soft bright timbres that are really missing from the DX family.

One additional note on the term FM as applied to Yamaha synths: no Yamaha synth uses FM (frequency modulation), they use phase modulation (PM) instead. There is a small but important difference in the underlying formula and in fact PM is more interesting musically, as real FM would always include strong timbre changes depending on absolute frequencies. To not confuse the reader - which I expect to care more for the musical rather than the mathematical context - I'll generally stick to the term FM for the Yamaha synths and correctly distinguish between FM and PM only in section 6, FM theory.

3.6 RCM (Realtime Convolution and Modulation) Synthesis

This is another marketing term by Yamaha and is mainly an extension to FM. The background is that you use a whole AWM2-element as modulator input for an AFM-operator, which also means that you can apply the filter on the sample before you put it through the FM-process. The former fact may be used to motivate the term modulation, the latter the term convolution (one possible algorithm for a filter is the convolution of the signal with a kernel). In my opinion the term RCM is ill defined and misleading. In lack of a better term, I will use RCM to denote the capability to feed the AWM section into the AFM section and vice versa.

More in section 5.1, RCM programming.

3.7 Phase Distortion (PD) and Interactive Phase Distortion (iPD) Synthesis

The terms phase distortion and interactive phase distortion were used by Casio for their synths (CZ and VZ series).

Actually the two methods are rather different. The phase distortion synths (the CZ series) offer eight basic waveforms (saw, pulse, resonance, etc) plus some combinations (e.g. saw and pulse). Each of them can be morphed continuously from and to a sine wave via an eight-stage envelope, thus emulating the use of a lowpass filter in a subtractive type synthesizer. Another two envelopes are available for pitch and amplitude. For two-oscillator patches, ring modulation and noise modulation is possible.[1]

[1] I will write about the VZs iPD as soon as I have hands on experience with it - I didn't fully understand the it so far.

3.8 Waveshaping

Waveshaping refers to a sound manipulation (not generation) technique which applies a (non-linear) function on the original signal (typically the output of an oscillator). This scheme is similar in principle to analog distortion in a guitar amp or fuzz unit, but offers much more sound variation possibilities, including e.g. resonance-like effects. This is achieved by complex and ideally dynamic waveshaping functions. Waveshaping can be used as an advanced synthesis method in a way similar to FM. In fact waveshaping has a lot of theoretical similarities to FM, but all the available products realize very little of its potential.

A rather simple waveshaping implementation can be found on some Korg synths like the T series and the 0/1 W. It can also be achieved on AFM synths in limited form (see section 5.6 Waveshaping Emulation).

3.9 LA (Linear Arithmetic) Synthesis

This buzzword was used by Roland to describe their approach to digital sound synthesis in the eighties. It is based on the observation that the attack transient of a sound is its most important part with respect to human perception. Therefore the LA-synths (D-50, D-20, D-10, MT-32 and others, but notably not the D-70) used a combination of sampled attack transients and simple digital oscillators with only sawtooth and pulse waveforms to generate the sustained part of the sound. (The D-50 also has a very limited set of looped waveforms that are mainly responsible for its typical breathy metallic timbres.)

The fact that those two parts were added or multiplied (also called "cross modulation") was used to motivate the term "linear arithmetic" - bogus terminology. The D-50 nevertheless offers an interesting synthesis engine, but I would say that the LA synthesis method as described above has little to do with its merits.

3.10 Ring Modulation, Amplitude Modulation

Ring modulation and amplitude modulation are not complete synthesis methods, but rather processing techniques that are quite common on advanced analog and digital synths. Sometimes these features are wrongly named or used when the actual implementation is quite different. Manufacturer specific terminology for similar schemes includes the terms cross modulation and FXM (frequency cross modulation).

Ring modulation is the multiplication of two signals. The output signal of a ring modulator will contain the sum and the difference of all available input frequency pairs.

Amplitude modulation is the multiplication of two signals, where one signal is always positive. In other words it can be produced by a simple level controllable amplifier.

3.11 Vector Synthesis

The first synth to implement this paradigm was the SCI Prophet VS. The VS can mix four oscillators with different waveforms in realtime via a joystick controller and a multistage envelope. While this is a really simple concept, it is effective for expressive play and nice evolving sounds.

The Korg Wavestations and the Yamaha SY22, TG33 and SY35 are other "vectorized" synths. The Yamahas can mix up to two FM and two sample elements, while the Wavestations mix up to four sample based wave sequencing oscillators.

In principle most synths can do realtime vector synthesis, when fed with MIDI joystick data to crossfade oscillators. (If you want to try that you can rewire a PC game joystick to fit your synths pedal jacks.)

3.12 Wavetable Synthesis

This term is used for two completely different things: Many companies that sell soundcards with RAM based sample playback capabilities use this term (because the samples are stored in a table in RAM).

For the PPG Wave and the Waldorf Microwave and Wave synths this term is used to describe the ability to produce a sound by sequencing through a table of different waveforms during the duration of a single note. For the wavetables and waves there is a preset ROM area as well as a user loadable RAM area provided. Which entry of the wavetable is selected may be controlled by an envelope, LFO or any other modulation source in realtime. Also these synths can interpolate between subsequent waveforms in the wavetable thus smoothing the timbral change. The waveforms are single cycle ones, so realistic acoustic emulations are out of reach for this technique, but the vastly improved modulation capabilities - compared to sample playback - more than make up for this.

3.13 Wave Sequencing

This term means that a sequence of different sample segments can be used to generate a sound. Korg implemented this on their famous Wavestation synths. The Wavestations oscillators can sequence through programmable patterns of samples. Each of the patterns consists of a number of individually tunable sample snippets and each sample in the sequence is assigned its own level and duration. Typical for the Wavestation (and rather easy to program) are "rhythmic" wavesequences in which an oscillator steps through a number of samples in a predefined periodic rhythm. The Wavestations also combine this with vector synthesis capabilities.

3.14 Granular Synthesis

Granular Synthesis means sequencing through very many very short sound (sample) snippets. The difference from wave sequencing is that the single samples are played for such a short time that the sequencing is heard more as a timbre than as a rhythm. Granular synthesis has been developed in the academic computer music scene and has not found its way into commercial products so far.

3.15 Linear Predictive Coding (LPC)

This is a synthesis method that has not found its way into commercial synthesizers, but was and is used in the academic computer music community especially for voice synthesis.

The idea is to base resynthesis not on a representation of the spectrum via Fourier coefficients, but to compute a filter that will respond to a pulse input with an output that matches the desired sound as closely as possible. Ideally a sound could then be represented by number of filters plus the remaining error signal. The latter can be expressed as a low resolution sample that is used as an additional input to the filter.

For the actual production of the sound one needs a rather simple input signal fed to a rather complex time varying filter. So effectively the difference to subtractive synthesis is merely the emphasis on resynthesizing the filter characteristics with great detail.

One nice property of LPC is that one can use a completely different filter input signal during synthesis than is required by the analysis. The result will share a lot of the timbral characteristics of the original, but may be e.g. pitched instead of unpitched. An important example would be to use spoken language as basis for the analysis, but to use an oscillator as filter input to produce a singing voice.

3.16 Impuls Synthesis, Voice Simulation (VOSIM)

I have found the two methods described here in different sources[4, 37] and without reference to each other. The approaches themself seem to be almost identical though. A simple form of Impuls Synthesis was implemented in one of the first wind synths, the German "Variophone", where it was called Pulsformung and allowed for expressive play of wind- and brass-like timbres.

Impuls Synthesis is a synthesis method that uses a parametrized pulse train oscillator as sound source, i.e. waveform consists of one or more pulses within one period. This oscillator can be controlled in pitch (cycle time), formant strength (number of pulses) and formant frequency (width of pulses) independently. This is of course a very useful property for vocal synthesis but also is said to allow for a natural vibrato of wind instrument sounds.

3.17 Physical Modeling Synthesis

Physical modeling (PM) is a whole class of synthesis methods. Traditional synthesis approaches are based on either an abstract mathematical description of sound - like the Fourier transform for additive synthesis - or on classical signal processing methods - like filtering for subtractive synthesis. Physical modeling on the other hand tries to mathematically model the diverse instruments themselves. The bow, string and resonant body of a cello or the picking finger, string and body for an acoustical guitar are described by mathematical means in a more or less complex form. In other words: while other synthesis methods only try to emulate the sound of a - real or imaginary - instrument, PM tries to simulate the creation process.

There are many different physical modeling algorithms including relatively simple ones like Karplus-Strong synthesis and rather complex ones like the waveguide approach which uses multiple delay lines to model strings or air columns.

The biggest advantage of physical modeling is the realtime control it offers. While other synthesis methods offer some algorithm specific and rather arbitrary control parameters like filter cutoff or modulation index, physical modeling enables the use of control parameters that are more musical and have a more complex influence on the timbre and phrasing. Examples for such parameters are embrochure or tonguing.

Another advantage of PM is that the sound generation is context sensitive: a note on a clarinet model will sound different if it is played with legato binding to its predecessor or with a little pause in between. This dependency is much more complex than with the traditional synthesizer portamento or glide function. Another example: the pitch bend of a saxophone patch will not just linearly shift the frequency of the note, but the synth will respond in a similar way to a real saxophone, i.e. it will shift the frequency and timbre for a while but then jump to the octave.

PM has its disadvantages though: Instrument models have to be designed with great care and a lot of knowledge of both instrument acoustics and the necessary math. On the available PM instruments by Korg and Yamaha, user editing is only possible via macro parameters of the otherwise hardcoded instrument models - probably the only practicable way to provide sound programming to the "normal" user.

Nevertheless, physical modelling is currently the only synthesis method that achieves the realism of sample playback without compromising the realtime control. In fact, it is exactly the realtime control that puts PM on the leading edge in realism since the artifical sound of a well done extensive sample set is mainly due to the lack of appropriate phrasing control.

3.18 Karplus-Strong Synthesis

This synthesis method uses a percussive sound - like a noise burst or a single pulse - which excites a delay unit with feedback. If the feedback level is high enough (90-99%) an exponentially decaying sound with definite pitch will result. It is the delay time that determines the pitch in this case. To be exact the delay time equals the period of the resulting periodical wave. This synthesis method is particularly well suited for emulating plucked strings and other percussive harmonic sounds. To make the decay more realistic, one can include a lowpass filter in the feedback path, so that higher harmonics are damped faster.

Karplus-Strong synthesis is actually a very simple form of physical modeling and shares the most important physical modeling advantage: since the percussive input sound acts as an exciter for the delay loop that produces the harmonic output sound, one can change the plucking/fingering of the modeled string by changing the percussive sound, and change the string properties by controlling the delay line parameters.

If you want to try this at home, it is possible with almost any delay unit. If there is no other way to push up the delay feedback close to 100%, you may also need a mixer with FX sends. Just patch the delay input to the FX sends and the delay output to a separate channel. The FX send of the delay fed channel now controls the feedback level. The additional advantage of this patching is that the mixers EQ can be used to influence the timbre.

Try a fast one-note tremolo with this simple setup. With a normal synth this will usually sound artificial, with Karplus-Strong synthesis it will sound like one is plucking a string repeatedly.

4 Basic Programming Techniques

Many people consider FM to be very hard to program or even a nightmare that a real musician should avoid to touch in order to retain his creativity. Well, I won't add a lengthy statement to this debate, let me just say that FM doesn't give you "instant gratification". If you are looking for some convincing nice sounds that you want to "just play", use analog synths or sample playback devices. Or collect a good sound library; an approach that works with any synthesis method.

There are *some* things that can be programmed fast and relatively easy, but I think that one will only successfully handle FM programming if one likes thinking about synthesis, analyzing sounds for their timbral characteristics and generally wants to learn about sound and sound perception - in a word: tries to understand what is going on rather than just tweak or fiddle around. If this is what you are interested in, try FM.

The main tool will always be your ears and your musical experience. Learning synthesis means developing a refined sense for timbre and learning to use the tools you have at your disposal to achieve your goals. With FM - as with any other of the more versatile synthesis methods - you will find that there are rules and standard ways to do things. I tried to list those that I know of below, but after you know how to apply them, it will be time to break them once in a while.

My personal opinion is that learning how to program FM is one of the best ways to learn about sound and synthesizers.

4.1 Some FM Terminology

Each AFM element contains of six *operators* which can be arranged in various ways by selecting an *FM algorithm* and setting up additional connections like e.g. feedback loops. An operator is just a collection of an oscillator with an envelope for amplitude control. The specialty of FM is that each operator has two inputs for audio range frequency modulation. By selecting the FM algorithm, one selects a basic patching of inputs and outputs of the six operators.

Some more terminology is helpful: those operators of which the output is directly used to produce sound are called *carriers*, while those that only modulate those carriers are called *modulators*. In graphical representations, the carriers are drawn in the bottom row of the algorithm scheme.

While there are 45 AFM algorithms plus more with external editing software, there are only a few basic setups of operators that will suffice 95% of the time. I tried to list the ones I find most useful and give a short discussion of their potential.

4.2 One Operator

Since it is more fun, I don't start with the best FM setup to explain, but with the one that gives the best immediate results.

Try a basic AFM patch with algorithm #1: Setup a feedback loop on operator 1 like in figure 1 and route the modulation wheel to EG bias, so that you can control the operator output level via the wheel. Set up amplitude modulation sensitivity to, say, 2 and eventually velocity sensitivity to about +3. Note that with this setting, an operator will output the level displayed in the output page only when the modulation wheel is full way up and you hit the

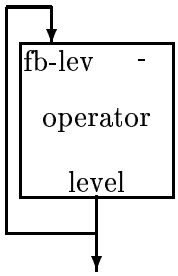

Figure 1: The FM setup that is the easiest to use: a single operator with feedback. Minimalistic but very useful.

keys with maximum velocity. Since, in this patch, operator 1 controls volume and timbre, the sound is controlled only by a few parameters, mainly the operator waveform, the envelope setting and the level.

One very useful timbre is a single sine operator with full feedback level and an output level of 115: This gives a nice sounding sawtooth wave which looks almost ideal on the scope too.

Play around with the various settings, especially with the envelope: try percussive, sustained, looped envelopes and envelopes with a short release peak. Play your sounds soft and hard employing velocity sensitivity and use the modulation wheel a lot.

For the sine wave as waveform, the operator level changes the sound from a sine wave (at low levels/feedback) to an ideal sawtooth (at level 115 for algorithm #1) and then adds aliasing "distortion" for higher levels. Other waveforms will distort at lower levels of course.

A little note concerning carrier levels in different algorithms here: Unfortunately, Yamaha decided to "normalize" the overall level for the different algorithms, and thus the maximum output level of the carriers varies depending on the number of carriers in an algorithm. This means that for feedback carriers, the level will have to differ for different algorithms to achieve the same sound. E.g. for the ideal sawtooth the level for algorithm #1 is 115, while it is 127 for algorithm #45.

4.3 2-Stack

This is the most important FM setup. The 3 feedback loops and the 16 operator waveforms on the SYs reduce the need for the "higher" stacks that were often necessary on the DX7 series.

In the following I listed the noteworthy special cases. While most books talk about frequency ratios in this context, I listed examples for actual frequency settings below. Thus, if you want to generate a spectrum an octave above the base frequency, you have to multiply all ratios by 2 etc pp.

Don't try to master all the following cases in a few minutes. You will need some time and experience to be able to get the effects you disire with each setting. Hear and learn.

Carrier Frequency = 1.0, Modulator Frequency = 1.0, eventually Feedback on the Modulator

This produces a harmonic spectrum with all harmonics present. For modulator levels around 100 and full feedback, the spectrum is very close to a linear sawtooth.

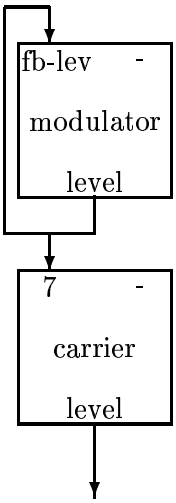

Figure 2: The FM workhorse 2-stack: two operators in series, possibly with feedback on the modulator.

Don't be confused by some books [18] that state a simple 2-stack with max levels and without feedback would produce a sawtooth-like sound. It does not, as is easily shown by comparing it to a real sawtooth. What Massey proposes - probably because of the shortage of feedback loops on the DX - produces a hard and somewhat unpleasant sound only very loosely resembling a sawtooth wave.

Carrier Frequency = 1.0, Modulator Frequency = 2.0, no Feedback

This produces a harmonic spectrum with only odd harmonics present. For modulator levels around 100, the spectrum is close to a square, Lower modulator levels make it closer to a triangle.

Carrier Frequency = 1.0, Modulator Frequency = high ratio, no Feedback

For modulator frequencies above the carrier frequency a harmonic spectrum with "gaps" of increasing size will result.

Combined with fast percussive modulator envelopes, the sound will resemble a "wooden" or "metallic" hit.

Very high modulator frequencies (around 31.0) will give very punchy "extra dry" bass sounds.

Carrier Frequency = high ratio, Modulator Frequency = 1.0, no Feedback

For not too high carrier frequency settings a sufficent modulator output level will still result in a clearly percieved fundamental. High carrier frequencies result in a pulsewave-like timbre.

Since the modulator envelope can control the level of the fundamental here, highpass filter effects can be emulated.

With medium attack times for both envelopes woodwind like timbres can be approached.

Carrier Frequency = high ratio, Modulator Frequency = different high ratio, no Feedback

This is often similar to the preceding setting in that the resulting timbre contains strong higher harmonics. The fundamental may be weak or completely absent, depending on the exact frequency ratio and the level settings (see section 6.1 Simple FM for details). Keep in mind that even with a completely absent fundamental frequency, one may still be able to recognize it.

Carrier Frequency = 0 Hz, Modulator Frequency = 1.0, eventually Feedback on the Modulator

The carrier acts as a "waveshaper". The resulting timbre is hollow, square-like for a carrier phase of 0 and will contain more even harmonics for phases around 32 or 96 (all this for a sine wave carrier). For details see section 5.6 Waveshaping Emulation.

Carrier Frequency = e.g. 1 Hz, Modulator Frequency = 1.0, Feedback on the Modulator = 7

The low frequency carrier adds a chorus effect to anything that gets to its inputs. In this example, the modulator will produce a sawtooth for high modulator levels and thus the sound of two detuned sawtooth oscillators will result.

Carrier Frequency = e.g. 300 Hz, Modulator Frequency = 1.0, no Feedback

This results in a partially harmonic, partially inharmonic timbre. Since the carrier frequency emphasizes a certain part of the spectrum, this setting may be used to emulate "formants".

For a voice or choir-like sound mix this setup with a more predominant harmonic sound component.

Carrier Frequency = 1.0, Modulator Frequency = e.g. 5000 Hz, no Feedback

This produces an inharmonic spectrum.

Can be used for metallic attack sounds like electric pianos and bells.

Carrier Frequency = 1.0, Modulator Frequency = e.g. 1.41, no Feedback

This setting produces an inharmonic spectrum.

Can be used for metallic attack sounds like electric pianos and bells. A modulator frequency ratio of 1.41 is what is said to be John Chowning's "favourite".

4.4 3-Stack

Of course this can be seen as an extension to the 2-stack which allows more complex modulating waveforms.

Less obvious applications of the 3-stack include:

- With the middle modulator in "LFO-mode" (frequency about 1 Hz, fixed), a chorus effect similar to a carrier LFO can be achieved.

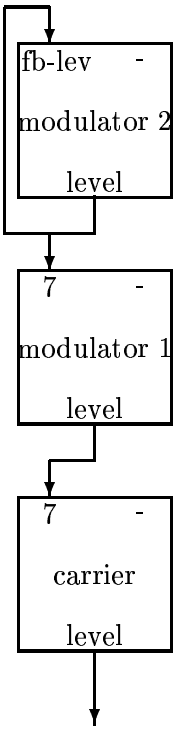

Figure 3: A 3-stack with feedback on the top modulator.

- Better square waves by using a 1:2:4 (c:m1:m2) frequency ratio.
- Good for very sharp metallic sounds like e.g. slap basses; use higher odd numbered c:m1 and c:m2 ratios.

4.5 2-in-1

Two simple FM 2-stacks where both modulators have exactly matching parameter settings may be replaced by a 1-in-2 configuration, which will produce exactly the same spectrum.

In the 2-in-1 configuration discussed here (see figure 4), the resulting spectrum is *not* a simple addition of two 2-stacks with equivalent parameters.

4.6 1-in-2

This configuration allows to use the same modulator to affect two carriers. This is nice to create thick timbres with few operators/feedback loops by detuning the two carriers or to generally merge 2-stacks that share the modulator settings.

4.7 "Long" Feedback Loops

I call a feedback from the carrier to a modulator in a stack a "long" feedback loop. This kind of setup will sound "distorted" and eventually equal a "fuzz tone" for high modulator levels and feedback.

If the carrier is set to low fixed frequency, say 1 Hz, a phasing/chorusing effect similar to the same setup with a simple 2-stack can be heard only it is a little deeper.

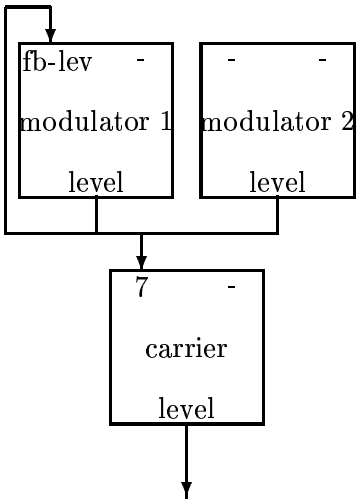

Figure 4: A 2-in-1 configuration with feedback on modulator 1.

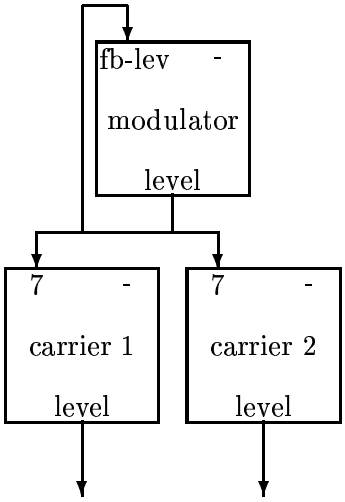

Figure 5: A 1-in-2 configuration with feedback on the modulator. Good for thick timbres.

Remember that for these long feedback loops, the carrier also acts as an modulator and thus the carrier level affects the timbre.

4.8 "High" Stacks

The AFM algorithm give you up to 6 operators stacked upon each other (in algorithm #1). This is not so very useful to produce complex spectra anymore, since the results become rather unpredictable. A very interesting application is using a high stack to produce a complex desynchronized sound animation by stacking up some LF-operators below a shorter stack that produces the basic timbre. Another interesting technique is to generate a complex waveshaping function by stacking up operators with different waveforms.

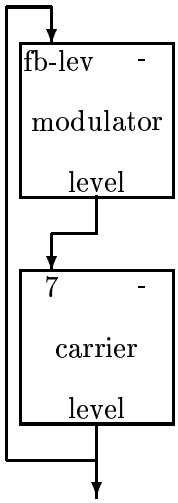

Figure 6: A 2-stack with "long" feedback loop. Ideal for "distorted" timbres and metallic voice sounds and for deep chorusing as well.

4.9 Some general remarks about FM programming

With "pure" FM, that is without combining carriers and without feedback, you will notice that FM produces a certain sound and not always a pleasant one. If you have a look at the Bessel functions that rule FM spectra, you will see that FM typically generates spectra with a lot of wiggle. It is as if the sound has a lot of resonances, only that the relative position of these is constant.

4.10 Beatings and Animation Techniques

One of the most basic programming techniques is to produce a beating which makes a sound fuller and eventually appearing to come from an ensemble of instruments. With FM the possibilities to produce beatings are more complex than with classical subtractive synthesis. I suggest to try the following hearing examples with both non-feedback 2-stacks (carrier and modulator output levels of 127) for a rather simple sound and richer sounding 2-stacks with full feedback on the modulator (modulator output level of 100). Turn off effects and everything else.

The standard way to produce a beating is to slightly detune two or more otherwise identical sounds. So detune two stacks against each other (say -1 for the first, +1 for the second stack): you will notice the typical flanging sound of two detuned oscillators. This already sounds very nice for the sawtooth-like wave.

Now take just one stack and detune the carrier and modulator against each other. The beating effect is much more subtle in case of the sawtooth waves because the carrier has little effect on the sound here anyway, but is more comparable to the two detunded stacks for the simple timbre.

...

4.11 Oscillator Phase

...

4.12 Envelope Programming

...

4.13 Additive Synthesis

...

4.14 Layering

...

4.15 Creative use of Samples

...

"That's the Waldorf MicroWave set for General MIDI."
"EEEEEIIIIAAAAAOOOOOOOUUUUMMMM
ZUMZUPZUMZUPZUMZUP..."
"Is that 'STRINGS 1'?"
"Yeah. You got a problem with that?"
Nick Rothwell

5 Advanced Programming Techniques

This section is a collection of programming techniques for the more adventurous people. Some of them seriously abuse the AFM section. Have fun.

5.1 Realtime Convolution and Modulation (RCM)

This somewhat misleading term is used here for a programming technique that routes an AWM element to an AFM element or vice versa (see also section 3.6). RCM is certainly not a breakthrough in synthesis, but it is at least a very valuable add-on to FM.

There are basically three kinds of RCM:

- The first one is using a sample as an input to the FM section, let's call that "plain RCM".
- The second - and less obvious - is using the sample section to modulate the FM section, but to use the FM section as sample source (i.e. instead of an actual sample). This may seem a little strange but has its uses, I'll call it "looped RCM".
- The third one is using the sample section merely as an additional amplifier/filter stage for the FM section, I'll call it "inverse RCM" and will tell you why I find it almost useless.

5.1.1 Plain RCM

I can see two somewhat distinct approaches to plain RCM programming:

- One may see the AWM section as a source for acoustic or otherwise hard to synthesize waveforms that are mixed, shaped and distorted with the AFM section. The most straightforward way would be to use the AFM operators at frequency 0 Hz and thus as waveshapers.
- Or one may see the AWM section as a kind of advanced FM operator in that it supplies more waveforms and its own filter and thus enables to extend the traditional programming techniques possible with Yamahas FM implementation.

RCM as extended sample playback For the first variety of plain RCM one important thing is to select the sample appropriately: Single cycle samples of course work just like any normal FM operator would. For more complex samples with long loops or timbral transients, it may be hard to achieve smooth timbre shifts. Thus it is best to use a sample that has (1) small dynamic variation and (2) not too strong resonances or beatings.

(1) holds since FM is very sensitive to modulator levels so you would get distortion for most noticeable RCM settings. You can also try to minimize unwanted distortions with the filter or amplifier envelopes or course.

(2) is since FM is most controllable with dull modulator waves anyway. Multisamples may be a problem as well since they usually include level and brightness changes. Using the AWM filters with high resonance or as highpass/bandpass also tends to distort the results.

Of course the above rules are only valid if you want to use the AWM input in a way similar to a normal modulator. Breaking them will just withdraw control from the programmer and tend to produce sharp resonances, aliasing noise and strong fluctuations and any of those may be desired.

Since FM can also be used for waveshaping (set the carrier frequency to 0 Hz, then the waveform selects the waveshaping function and the phase sets the function's origin), one nice use of RCM is for distorting samples (see section 5.6 waveshaping).

For practical use of plain RCM besides using basic single cycle samples like triangle, try synthetic waves like synth choirs. RCM adds some grunge/hiss and makes them more lively overall.

One nice setup is having a sample modulating all 6 FM operators in parallel (algorithm #45) and setting the operators frequencies to say 1.0, $1.0+\epsilon$, $1.0-\epsilon$, $2.0+\epsilon$, $2.0-\epsilon$, 3.0, i.e. to the fundamental and the first harmonics with slight detuning. This will result in distorted and/or frequency shifted copies of the sample which can be more or less subtly mixed with the original for a more lively sound.

Another possibility is to use the AWM filter section in 24 dB lowpass mode with resonance turned up to the point of self-oscillation, but to shut off the direct AWM output and thus control the oscillation by means of the AFM operator envelopes.

One additional hint if you want to use plain RCM as a way to use complex or steep filter effects. When using one carrier operator with zero frequency and AWM input you can get an almost linear waveshaping with distortion only for high AWM levels. Try this with a resonant lowpass filter in the AWM element and you get a type of distortion that at least loosely resembles certain kinds of filter distortion in analog synths. If you want more resonance you may use the AFM filter too, but try using a bandpass configuration instead of a resonant lowpass here, it helps for that famous plastic sound.

RCM as extended FM There are some really unique possibilities in using the AWM section as a kind of extended operator. If you have a look at what kind of FM "patches" were used in the academic computer music scene [8, 3, 32, 23, 35], you will find that many of them are not possible to reproduce with Yamahas FM implementation. This is because there is no way to specify separate carrier and modulator frequency ratios or different envelopes for different notes. The most important deficiency seems to be the lack of operator frequency scaling which is important for e.g. the typical frequency spreading of higher partials on the piano or the generation of formants.

With RCM one can approach these programming problems by using e.g. microtuning tables for the AWM "operator" only or by filtering a sawtooth wave with a narrow bandpass at a constant frequency.

When creating "weird" sounds, one can also employ more drastic effects by using excessive filtering on bright AWM waves at high frequencies.

5.1.2 Looped RCM

Concerning "looped RCM", one interesting thing is of course to control an FM feedback loop directly via the amplitude modulation or filter cutoff modulation of the AWM Element. The

problem again is that the sum output of the AFM Element is used for the feedback (instead of a single FM-operator or stack), so to avoid distortion you need to avoid unwanted level changes - most notably beatings! Thus it works best with very simple FM algorithms using only a few operators.

5.1.3 Inverse RCM

As described, inverse RCM means simply running an AFM element through an AWM element. Of all the AWM setting only the amplitude envelope and the filter section have any effect here. Nevertheless, this mode could also be named "almost useless RCM" since the Yamaha engineers in their eternal wisdom choose to route the AFM signal *before* the filter through the AWM section. So if you shut off the output of the AFM element, the AFM filter will have no effect at all - forget about 48 dB lowpass, 24 dB bandpass, bandpass with resonance etc. - all you really get is an additional amplitude section with envelope and LFO. However, note that the AFM signal including filtering is always available at the normal AFM output as usual.

The only way to come close to the aforementioned filter effects is via plain RCM will almost linear waveshaping.

5.2 User Defined Algorithms

The SYs allow for user defined FM algorithms by using a so called "free algorithm edit" via external MIDI software. The Emagic SoundDiver SY77/99 application I use implements this in a rather nice fashion and also adds the necessary documentation. User defined algorithms are a complex matter, since you directly assign the various registers and there are limitations in the number and kinds of connection.

One useful extension to the 45 preset algorithms is what I call "algorithm #46", which is nice for very fat-sounding patches with some complex FM-ing on top (see figure 7).

5.3 Multisample Shifting

Unfortunately the SYs do not have a parameter for shifting multisamples so that the pitch remains unchanged, but the timbre is shifted. You can however gain this by using the pitch EG and/or the LFO. Set all pitch EG levels to maximum with a PEG range of 2 octaves, which will result in the sample being transposed 1 octave up. Then go to the voice common parameters and select an element transposition of -12. This will result in shifting the multisample 1 octave. For the other direction set the transposition to -12 and the pitch EG levels to minimum.

A similar effect is possible using the LFO with frequency 0, square waveform, maximum sensitivity and a phase of either 0 or 50. In this case the sound will be slightly detuned though (see section 5.8 LFO emulation).

5.4 Realtime Control of Detuning

Unlike some early Casio synths, the SY series does not offer realtime control of detuning via a MIDI controller. This is a pity because controlling the detuning effect dynamically with e.g. a modulation wheel adds considerably to lively play. Here is a way to get it nevertheless by sacrificing the main LFO.

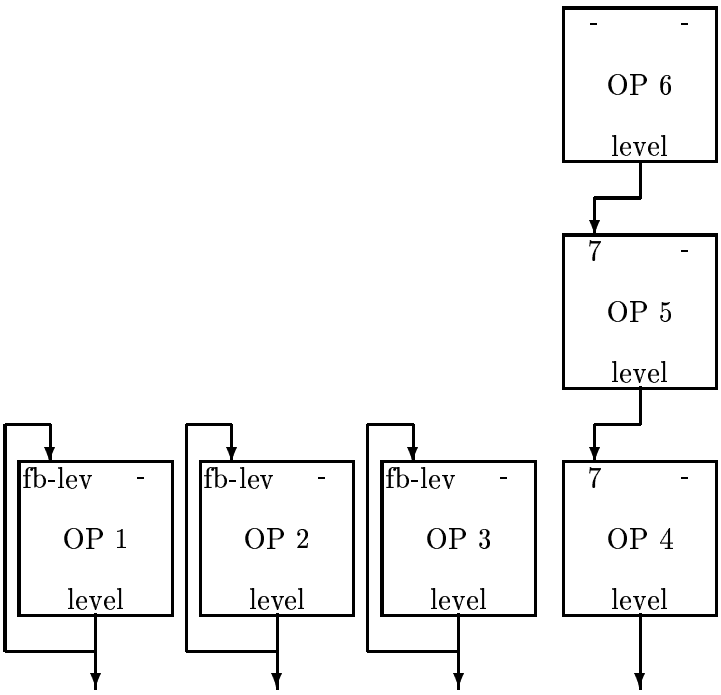

Figure 7: The "4 oscillator" algorithm which I call #46. This one is only obtainable by external software via "free algorithm edit". Nice are the 3 "oscillators" that may be used for some "meat", while the 3-stack to the right can be used to add metallic attacks and other more complex FM specialties.

Set the LFO waveform of the element you want to detune to square, the LFO frequency to 0 and the wired in pitch mod depth to 0, but assign a realtime MIDI controller, e.g. the modulation wheel, to add pitch modulation with reasonable depth. Now you can use the modulation wheel in realtime to control the detuning. If you want to detune a two element voice symmetrically, set the LFO phase of one of the elements to 50 so that the LFO starts with the negative half cycle. The duration of the half cycle for this setting is about 16.5 seconds, so you may notice a sudden pitch shift if you play *very* long notes.

5.5 Velocity Sensitive Detuning

Another feature missing in the SY series architecture is velocity sensitive detuning. But again sacrificing something - in this case the pitch envelope - helps. For two elements to be detuned by increasing velocity, set up the pitch envelopes so that all levels are slightly positive for element 1 and slightly negative for element 2. Try 1/2 octave pitch range and levels of say +5 and -5 and switch on the velocity sensitivity of both pitch envelopes.

A similar effect is possible for different stacks in an AFM element by switching the pitch envelope off for one stack.

5.6 Waveshaping

Waveshaping is a synthesis method which uses a nonlinear function - the waveshaping function - to "distort" an input wave. This can be done using a special operator setting on the SY

FM-synths.

If one writes down the FM equation

$$A(t) = w_c[2\pi f_c t + \phi_c + I w_m(2\pi f_m t)]$$

where

t	time
$A(t)$	output signal
$w_c(t)$, $w_m(t)$	waveform-functions of carrier and modulator
I	modulation index
f_c, f_m	frequencies of carrier and modulator
ϕ_c	carrier phase

it is easy to see that by setting the carrier frequency f_c to 0 the carrier waveform-function becomes just a kind of transfer function with a parameter ϕ_c that shifts its offset:

$$A(t) = w_c[\phi_c + I w_m(2\pi f_m t)]$$

So if e.g. the carrier waveform is an identity function (sawtooth or $\sin(x)$ for small $|x|$), you will get the raw modulator as output and if the carrier waveform is nonlinear, the phase will shift the transfer function.

Waveshaping is a rather general synthesis approach that can be mathematically analyzed in a way similar to FM. One important result of this analysis is that for a sine input, an even waveshaping function will result in a spectrum consisting of only even harmonics, while an odd waveshaping function will produce only odd harmonics. Both of these extrema as well as their mixtures can be achieved by using e.g. a sine waveshape with phase set to 0 (odd function) or 32 (cosine = even function)

So much about the theory, but what is this good for soundwise?

Take a 1 AFM + 1 AWM voice, with the AWM element programmed as e.g. a clean Stratocaster guitar sound (on the SY99 try the first element of preset "PL:Caster"). Now switch off the direct AWM output by setting "Voice Common - Output Group - AWM" to "off" and initialize the AFM element. Then set the frequency of operator 1 to 0 Hz (fixed) and its external input to AWM with level 7. What you have programmed here is a RCM voice: the AWM section is solely routed through operator 1.

The sound you get should be very similar to the AWM-only guitar sound.

Try changing the phase of operator 1: around 32 and 96 the sound will be lower volume and "distorted", while around 0/127 and 64 it will be almost identical to the original AWM sound.

Other waveforms for operator 1 will give you more distorted sounds or even no sound at all (try waveform #3 with phase>64). Also the input level of operator 1 will affect the sound. The waveform and (in case of low input level) the phase of operator 1 determines the "waveshaping function", while at least for waveforms which are linear for small input (#1, #13, #16) the input level will control the waveshaping "amount".

So basically I can think of the following applications:

- "Distort" sounds depending on their volume.
- Create complex distortion/chorusing effects by routing an AWM element to modulate all six operators in algorithm #45 in parallel, but with different waveform settings and eventually frequencies in the 0.1 to 1 Hz region. Try this for the clean guitar example!

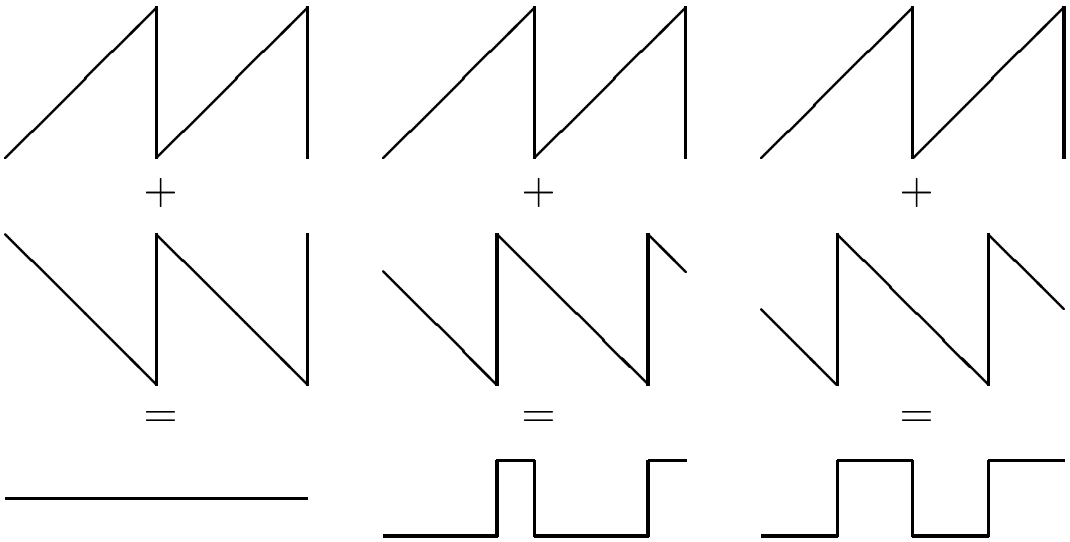

Figure 8: How to get PWM as a superposition of a positive and a negative sawtooth wave with differing phase. In the left column, a sawtooth and a negative sawtooth are summed up to a zero signal. In the middle column, a sawtooth and a negative sawtooth shifted by 90° are summed up to a 25% pulse wave. In the right column, a sawtooth and a negative sawtooth shifted by 180° are summed up to a square wave (a 50% pulse wave).

- Simulate resonating sweeps by selecting something like wave #4 and setting the frequency of operator 1 to around 0.1 Hz.
- Get repetitive sounds with waves like #3 with small fixed frequency.

5.7 Pulse Width Modulation

Pulse width modulation (PWM) is one of the nicest effects on most analog synths - and of course a well-know cliche. PWM means that the pulse waveform of an oscillator is modulatable in its pulse width by some external source like an LFO or an envelope. Unfortunately most digital synths do not offer "real" PWM, eventually you will find samples for some of the standard settings - how boring!

There is however a really nice trick[2] to get the real thing: Imagine you have two sawtooth waves the second one being the negative of the first. If you add both of them, you get a zero signal for no phase difference or a square wave for a 180° phase shift or a pulse wave for something in between, see figure 8.

One possibility to realize this is with AWM: AnalogSaw1 and AnalogSaw2 are exactly the same sample but in original and negative form (a triple hooray to the Yamaha sound programmers for this one!). So set up a "2 AWM" voice and select AnalogSaw1 and AnalogSaw2 as sample waveforms. Unfortunately the AWM elements have no phase parameter, and if you check out this setting you will see that such a parameter wouldn't make much sense: the different elements of a voice are not phase synchronous, therefore the actual pulse width you get with the above setting depends on the key you play (there seems to be an almost constant delay between elements, resulting in a pitch dependent phase shift). Nevertheless the resulting sound is very usable.

[2] I read the basic idea for this in an article by Jörg Holzamer in the German KEYS magazine.

There is something more in PWM by AWM though: By using the Pitch EG on one of the AWM elements, with the first level (L0) set to about +15, the rate to about 45 and the velocity sensing to +7, you can control the pulse width by velocity.

Another possibility is to use the LFO on one AWM waveform: set the LFO waveform to square, LFO frequency and pitch mod depth to 0, and assign the modulation wheel to pitch modulation. Now you can use the modulation wheel in realtime to control the pulse width mod speed from zero to quite fast. In fact this LFO trick basically sets up a realtime modulatable detuning, which the SYs otherwise don't offer (see also section 5.4 Realtime Control of Detuning).

So that's all about "real" PWM on SYs? Of course not! The AFM section again offers much better possibilities: set up an AFM element with say Algorithm #32 (two 3-stacks) and program two identical 2-stacks with frequencies = 1.0 and maybe some feedback on the modulator to get a nice sawtooth wave for each 2-stack. Now set the carrier phase of the second stack to be ±180° of the first (i.e. phase = 0 on the first carrier and phase = 64 on the second carrier). This will - for any otherwise identical stacks - result in inverting the waveform und thus silence because of complete cancellation. You can get PWM-like effects for any basic waveform and - by using the same amplitude envelope settings on both stacks - even for evolving ones! The only exception to this rule are stacks which use the carrier as feedback source, because in this case the whole waveform is just shifted in phase - not inverted.

But for our special example with sawtooth-like stacks we get the same effect as with the 2 AWM voice described above. The big difference in using AFM instead of AWM is that the AFM operators may be synced and have a phase parameter. Thus you can use the Pitch EG, the LFO or the other operators for any kind of additional modulation.

A little note though: when using "perfect" phase cancellation you'll get quite imperfect results when playing the same note repeatedly: it seems the phase initialization is nearly but not quite correct!

One more extension to the PWM trick: Imagine 3 sawtooth waves each 120° apart. This is a sawtooth tuned a factor 3 above (which is equivalent to a musical 12th). Now detune saw 1 down and saw 2 up a little: continuous sweeping between fundamental saw, 3rd harmonic saw and 2nd harmonic saw in between.

And the really last one: The most "perfect" square wave I could get with 3 operators is by using Algorithm 34 with feedback = 7 on operator 3, and levels 127, 94, and 94 for operators 1 to 3. This square looks a little slanted but has nice sharp corners. Now take two of those stacks with 180° phase difference for the carriers and you get a nice "± pulse width modulation"

5.8 More LFOs

If you find that you need more than the 1 1/2 LFOs included in the AFM section, sacrifice an operator instead. You can get a LFO with 16 waveforms by setting one of the modulators to a low fixed frequency. And more if you use e.g. the sine waveform with phase 32 and loop the operators envelope to achieve the desired waveform.

There are two disadvantages though: first the maximum pitch shift is rather small and second it is not constant (in terms of musical intervals) for different frequencies.

You will find that at C3 (ca. 261 Hz) you get a frequency shift of ± 1 semitone with full modulation level. This frequency shift will double when going one octave down and half for each octave up.

As a starting point, set the modulator level to 127 and set the level scaling to about -70 for the lowest note and to 0 for the highest note on the keyboard.

See also section 5.9 More Pitch Envelopes.

5.9 More Pitch Envelopes

Similar to the LFO trick described above, you can also get additional pitch envelopes. Set a modulator to the fixed frequency 0 Hz and its phase to 32 for the sine waveform. This operator will now produce a constant positive value at its output - for a negative value you may use phase = 96. And voila, the amplitude envelope influences the pitch of the selected carriers. However if you have a close look at the FM formula (see section 6) you will see that the modulator envelope does not act as a pitch envelope, but as a phase envelope. This means that the pitch shift is equivalent to the derivative (slant) of the modulator envelope. You will also note that - since the SY uses exponential envelopes - the shift is not proportional to the slant of the envelope you see in the LCD graphics, but varies even *within* one envelope segment.

The same disadvantages as with the LFO emulation are present, so again set the modulator level to 127 and set the level scaling to about -70 for the lowest note and to 0 for the highest note on the keyboard.

5.10 Resonance Modulation

The SYs do not offer modulatable resonance in the filter section. If you ever used a MicroWave or another synth with this feature, you will certainly miss it.

One variety of resonance modulation is available though with a little trick: Select LPF for both filters, so that you have two independent 12db lowpass filters at hand. Set the cutoff frequency, the frequency scaling and the envelopes to identical values for both filters and set the resonance to some high value - even 99 is fine. For resonance levels above 83 or so you might get that ugly self-oscillation - don't worry we'll handle that. If you slowly change one of the cutoff frequencies, you will notice, that the resonance decreases and even with resonance = 99 the self-oscillation will disappear. You can use this to change the resonance over time by editing the two LPF envelopes, so that the cutoff frequencies come closer and withdraw again. A short highly resonant attack phase followed by a lower resonance sustain phase can make sound quite punchy - check it out.

Another possibility is setting one filter to LFO control and the other one the EG or EG-VA control. By this, you can change the resonance in realtime via a MIDI controller, e.g. a modulation wheel.

5.11 Amplitude Modulation

The AFM amplitude envelopes are very fast and also loopable. So try to use a looped envelope with highest rate settings (63) on the looped part and level changes of only 1 between adjacent levels. The result is a small but fast amplitude modulation.

5.12 Effects Programming

There are of course some FX settings that are more or less obvious and generally useful like the all present Chorus→Reverb combination. So I will just discuss some more exotic and SY

specific FX setups. Note that all of the following applies only to the SY99.

5.12.1 Reverb

The Plate Reverb is able to add a considerable metallic flavor to almost any sound - even with short reverb times around 1 second.

The Early Reflection algorithms and especially the Gated Reverb sound extremely artificial and are well suited to seriously warp or redefine a timbre. Try these and other reverb algorithms with very short time and delay settings and lots of diffusion and "Liveness".

5.12.2 Distortion

I find the distortion algorithm mainly useful for high gain sounds. It sounds like a really bad amp if you get the most out of it. Of course it is nice for techno type sounds and especially bad sounding kicks.

For other distortive desires consider using waveshaping (see section 3.8).

One nice exception from the rule is adding slight distortion (say 10% and not with too much treble) to bright "analog" sounds. The effect - if not overemphasized - is to make the sound more "buzzy" and shadow the sound character of the SY.

A very slight distortion or exciter type effect can be achieved by means of the EQ part in some of the algorithms. A high band boost doesn't have much effect on the sound at high frequency settings (try to hear a difference at 16kHz), but is able to add a grainy boost at 500Hz (not enough precision in the DSP algorithm?). Try boosting with the high band at 500 Hz, the mid band at 6.3 kHz and the low band at 2 kHz. This gives not only a volume and EQ boost but also a very nice and subtle distortion.

5.12.3 Resonators

The FX section may also be used as a resonator. This can simulate e.g. the corpus of an acoustic guitar or - more generally - add consistency, liveness and character to a sound.

The basic idea is to set up one or several fed back delays or resonating filters so that some frequencies will be emphasized during the sustain and decay phases of a note. The biggest advantage of this is that the timbre of the instrument will become contextually sensitive: how a note sounds is not only defined by its velocity, controllers, etc. but also by what the last notes played were sounding like. A very musical property that is otherwise only achieved via physical modeling synthesis.

While the SY99 FX section unfortunately does not offer a dedicated resonator algorithm, any algorithm that does include a fed back delay line or a resonant filter will do as a substitute. This includes delays, some reverbs (e.g. Early Reflection), some modulation algorithms (e.g. Flanger and Phaser), the pitch shifters 1 and 2, the "Wah"-effect and the exciters. Most useful are those algorithms that allow for several feedbacks with independent delay times, since the resulting resonator will be more complex.

5.12.4 Modulated FX

By using one of the modulation algorithms - i.e. phaser, flanger, chorus, symphonic or rotating speaker - in the FX section of the SY99 you can get effects similar to ring modulation or similar techniques.

Set the FX modulation destination to the frequency parameters of the algorithm and select e.g. velocity with full positive depth as modulation source. This gives you a velocity sensitive shifting from e.g. soft chorusing to metallic distorted sounds typically associated with ring modulators.

6 FM Theory

6.1 Simple FM

The following is the basic form of "AFM" as implemented in Yamaha synths.

A simple "2-stack", i.e. a carrier-modulator-pair looks like this:

$$A(t) = w_c[2\pi f_c t + \phi_c + I w_m(2\pi f_m t + \phi_m)] \tag{1}$$

where

$$I = \frac{\Delta f}{f_m} \tag{2}$$

The symbols used are:

t	time [seconds]
$A(t)$	resulting amplitude signal
f_c	carrier frequency [Hertz]
f_m	modulator frequency [Hertz]
I	modulation index
$w_c(t)$	carrier waveform (2π-periodic)
$w_m(t)$	modulator waveform (2π-periodic)
ϕ_c	carrier phase [rad]
ϕ_m	modulator phase [rad]

This is already the extended form of FM as used in the SY series. In the following we will simplify this by setting the initial phases to 0 and both waveform functions to sinewaves, thus:

$$A(t) = \sin[2\pi f_c t + I \sin(2\pi f_m t)] \tag{3}$$

This will generate the frequencies:

$$f^{(k)} = f_c \pm k f_m \quad \text{for } k = 0, 1, 2, 3, \ldots, b \tag{4}$$

where b is the approximate bandwidth of the FM spectrum, which may be estimated by

$$b \simeq I + 2 \tag{5}$$

The frequencies $f^{(k)}$ are called the k-order sidebands. Their amplitudes are determined by the respective Bessel functions $J_k(I)$. The resulting signal for simple sinewave FM can thus be calculated as

$$A(t) = a_c[J_0(I)sin(\phi_c t) + \sum_{k=1}^{b} J_k(I)sin((\phi_c + k\phi_m)t) + (-1)^k sin((\phi_c - k\phi_m)t)] \tag{6}$$

where a_c is the carrier amplitude.

6.2 Complex FM

For the general case of the modulator having a complex spectrum, one gets:

$$A(t) = a_c \sum_{k_1=1}^{b_1} \sum_{k_2=1}^{b_2} \cdots \sum_{k_n=1}^{b_n} J_{k_1}(I_1)J_{k_2}(I_2)\cdots J_{k_n}(I_n)sin(C_n + \sum_{i=1}^{n} k_i\phi_i)t) \tag{7}$$

where n is the number of sinewave components in the modulator spectrum.

7 Realtime Control

Realtime control via wheels, sliders, pedals, aftertouch, breath controllers, ribbons etc. is one of the strongest points of FM synthesis. This is because FM synthesis offers control parameters like the level or frequency of a modulator in a FM algorithm that are unique and can result in a versatile range of sound changes. Unfortunately the control capabilities of the AFM section in the SY synthesizers do not even match the one on an old DX7 II - especially with the E! extension - where not only the operator level, but also the operator frequency, the envelope and - with E! - the level of all modulators can be controlled via MIDI. That is not to say that the modulation capabilities of the SY series are weak in general - in fact they are better than on most other synths - but they have some serious and perhaps unnecessary limitations. Some tricks to make things a little better are included in the following descriptions.

If you found controller programming too complicated and not really useful so far, give it another try. I propose to program one or more editing templates which set a lot of controllers to useful defaults, following the guidelines given in the following paragraphs.

7.1 Pitch Bend

Pitch is certainly the most important control parameter for musical sounds. The SYs have pitch control hard wired to the pitch bender with just a simple range parameter, no inversion, no quantization (like e.g. on the DX7II), no key modes (at first glance), no nonlinear settings, and only an octave as maximum amount.

It gets a little better with the aftertouch though, since it has a separate pitch bend parameter. What is especially nice about aftertouch pitch bend is that it is affected by the aftertouch mode, i.e. can be set to bend only the highest or lowest note played or be restricted to a certain key range. The amazing - and as usually undocumented - feature is that the aftertouch mode also affects the pitch bender, i.e. if aftertouch mode is set to "top" the pitch bender will shift only the highest note played. By the way, the aftertouch mode does also affect all other aftertouch controller routings and thus is ideal for e.g. playing an expressive solo line on top of some chords: with aftertouch mode "top", you can assign aftertouch controlled vibrato or filter modulation to the solo voice only.

7.2 Breath Controller

The breath controller in its actual realization, the BC-2, is a nice little device that resembles a head mounted wireless microphone with a small mouthpiece attached to a flexible wire. It takes some training to get the necessary endurance in blowing, but the breath controller excels as a control device in speed, ease of phrasing and practical dynamic range. Try playing sforzato, legato, staccato and sudden stops via tonguing. Also try slow swells and fades and humming or singing into the mouthpiece. Use all these playing techniques in one melodic line and you get something you'll never be able to do on a keyboard. Mastering all this will take considerable time and effort, but is well worth it.

The usual - and best - way to employ a breath controller is to let it control the volume of the sound with full range, i.e. from silence to fortissimo. Thus, to instantly make a sound playable with a breath controller, set the "volume low limit" to 0 and assign MIDI controller #2 (breath) on the controller page. Now you can play the synth in a melodica fashion: there will be no sound unless you blow into the mouthpiece.

To really play the synth like a wind or brass instrument though, you have to program special patches. If you want to play the patch polyphonic use the envelopes sparingly or not at all. You may program some pitch instability (via pitch EG) or other attack noises, but this will disable playing real legato style, which is one of the biggest advantages in using the breath controller. For monophonic play of AFM-only patches you can use one of the mono voice modes with fingered portamento alternatively. This setting disables the envelopes for legato play and thus gives you more sound variation for different phrasings.

It is very important to assign additional parameters to the blowing pressure, so that there is a significant timbral change over the dynamic range. Some classical settings are:

- If BC is assigned to EG bias, which controls the AFM modulators (you will have to set the amplitude modulation sensitivity), the resulting timbre gets brighter with high blow pressure.
- An effect similar to overblowing into e.g. the octave can be realized by using one AFM stack or element for the fundamental and programming another stack or element one octave higher with high positive EG bias for amplitude modulation. This works best with FM stacks, since assigning the modulation to both carrier and modulators will result in a breath response that is most prominent near maximum pressure.
- BC assigned to EG bias for a 2 AWM voice, where amplitude modulation is positive for one element and negative for the other. Thus you get a crossfade between the elements.
- BC assigned to cutoff frequency, where modulation is positive for a lowpass or bandpass filter or negative for a highpass filter. This is a kind of brightness control too.
- BC may also be assigned to pitch and/or amplitude modulation via the LFO.
- BC routed to pitch modulation by a zero frequency square wave LFO can assign a slight frequency shift to the blow pressure. Try to adjust it so that a medium pressure is tuned perfectly, while high pressure makes the sound sharp and low pressure makes it flat. Of course the sound will only be playable as pitched voice, if the controller depth is very low.
- BC may be routed to FX parameters on the SY99 or possibly on external FX units. Obvious assignments are for reducing reverb depth on high pressure or for increasing the modulation depth of chorus effects.
- Assigning BC to both volume low limit and EG bias results in an exponential controller characteristic.

When playing with the breath controller, don't forget aftertouch, portamento (finger controlled with a monophonic AFM voice or pedal switch controlled) and other controllers.

Nice control parameters that are only possible via sysex routing on the SYs are: portamento time, operator frequency (for overblow effects) and filter resonance.

7.3 Panning

The panning section is a bit obscure on the SYs, there are relevant parameters on the controller pages, on the pan page and in the separate pan tables. A good template to start with would

use MIDI controller #10 assigned with full depth to "Pan bias", some other controller assigned with full depth to "Pan LFO" and the Pan set to some preset. Note that pan can be controlled by either LFO, its own loopable envelope, note number or velocity, but not by more than one in parallel.

7.4 Filter Control

The filter control is a bit counterintuitive to program the first time. The LFO depth parameter on the filter page does not only control LFO modulation depth, but also the controller depth if a controller is assigned on the controller pages. Furthermore there is a big difference between the LFO mode and the EG and EG-VA modes. In LFO mode you gain realtime control of the filter frequency via the controller, while in EG and EG-VA modes the controller is sampled on note on and then will stay constant for this note. Both controller modes are of course useful, although it is a pity that realtime control is not possible while simultaneously using the filter envelopes. A standard setting for filter control would be a +3 LFO depth with full range assignment of the 2nd wheel (#13) to cutoff frequency. If you want a subtle timbre modulation try assigning the LFO with positive depth to amplitude and with negative depth to the cutoff frequency of a lowpass filter so that the overall loudness stays approximately constant.

7.5 Effects Control

The effects control section on the SY99 is a very nice thing, although it has some drawbacks:

- If a controller is assigned to an effects parameter, the parameter setting is initialized to zero instead of the value given in the parameter setting.
- There is considerable zipper noise on almost all effects parameters, at least if you use the FX LFO.
- There is no way to synchronize the LFOs of the different effects.

A minimal use of the FX control section is to set up one or two FX parameters to be controlled by a wheel, pedal or slider. For instance using 2 sliders on a fader box as standard FX controllers, one can get easy control of, say, reverb and chorus depth.

7.6 Realtime SysEx Control

There is a way to compensate for the limited modulation capabilities of the SY series if you use a computer with appropriate MIDI software and if you only want to modulate one voice in parallel. The SY77, TG77 and SY99 support realtime parameter changes via MIDI system exclusive (sysex) messages. If the synth receives such a message it will go into voice edit mode and change any sound parameter in realtime.

Most of the better MIDI sequencers like Cubase or Logic support mapping of usual MIDI controller messages - e.g. modulation wheel movements - to definable sysex strings. Thus you will have to control the synth via MIDI routed through the computer. Another possibility is to use a slider box like the Peavey PC-1600 that is able to assign sysex messages to sliders, buttons and control inputs. Some really useful target parameters are the frequency of AFM operators, envelope rates and levels, filter resonance, LFO frequency, portamento time and aftertouch pitch sensitivity.

However, remember the disadvantages of this method:

- Can only be used for one voice
- Changes apply to all notes
- A lot of MIDI bandwidth is needed
- Problems with program changes: the SY will ask if it shall store the edited voice.
- You need a computer with a dedicated software setup or another hardware device (fader-box, MIDI patchbay) connected

For advanced use of sysex and serious MIDI processing, there is one software package that is especially suited: MAX[3].

MAX is a programming language with a graphical editing interface that is available for NeXT workstations with ISPW hardware or as a commercial software package for the Apple Macintosh from Opcode. MAX covers all kind of MIDI realtime processing and includes an "sxformat" object that allows you to easily specify the system exclusive templates that you want to transmit. When the sxformat object receives a number, e.g. a MIDI continuous controller value from a slider box, it will insert that number (or some MAX-calculated variant of it) into the placeholder slot in the template and transmit the entire sysex message. The sxformat object can be triggered by any kind of MAX event, which includes all kinds of MIDI messages as well as key hits on the computer keyboard, MAX internal timers, etc.

[3]Thanks to Ken Beesley for the following information

8 Emulation of Instruments

This section is a collection of hints and programming techniques for instrument emulation. Don't take anything too serious but play with the ideas - not only to emulate existing instruments. Wendy Carlos suggested [6] to programm a library of all orchestral instruments as a good synthesizing learning exercise. In fact she programmed several hundered patches for early additive synthesizers. I'm not sure if it is wise to go that far, but trying to emulate at least the most important instrument families seems to be mandatory for a synthesist to me. Also try to manually "copy" sounds you hear or have available on other synths. This is by no means waste of time but a very good exercise to analyze a given sound only by hearing. The advantage is that you *must* be able to come really close to the desired result if the synth used is not completely different from the one you program.

8.1 Brass

The spectrum is sawtooth like with "metallic" resonances. The attack phase contains pitch and timbre instabilities.

The classical setup for a subtractive synth is sawtooth wave into a 24 dB lowpass filter with a filter ADSR (fast attack, medium decay and release, high sustain level). The instabilities can be modeled by means of a pitch envelope (very fast attack with low→high→correct pitch), an envelope controlled LFO (on the SY use the Sub LFO in decay mode with short time setting and high LFO frequency) and eventually the filter and FM modulator envelope (similar setting to the pitch envelope). A "Plate Reverb" can add considerably to the metallic sound. Good control techniques are of course using BC or key velocity to control the brightness. A pedal operated damper effect may be nice too.

8.2 Bowed Strings

The spectrum of a single bowed string is sawtooth like with a slow attack and decay. A minor but eventually important component of the timbre is the "bow" sound which is noise like. At the beginning of the attack phase the sound may be slightly flat. There are characteristic resonances for the different members of the instrument family. An important phrasing characteristic is the the alternating up/down bowing, where the up bow phase has a "harder" attack. Delayed vibrato is usual. A string ensemble is defined by a very dense chorusing.

Classical setting for a subtractive synth is a sawtooth or a medium width pulse wave into a 12 dB lowpass filter and eventually a resonator (zero frequency chorus or parametric EQ). For the bow sound, a little white noise may be added. For the string section as many oscillators as possible are used with lots of detuning and eventually pulse width modulation and differing vibrato rates.

For FM programming, all variants of sawtooth setups and the chorus effect gained with low frequency carriers in 2-stacks or the middle modulator in 3-stacks may be used. The "Symphonic" effects algorithm of the SY99 does a good job on "diffusing" a string ensemble.

For solo strings a more uneven resonating spectrum can add realism. This can be achieved by using FM with little or no feedback (no buzzing sawtooth this time).

Besides controlling the brightness by key velocity or a pedal the up down bowing may be simulated by a footswitch. A lot of the special playing techniques like tremolo, pizzicato or flageolett may be emulated by the use of switches.

8.3 Choir

The spectrum is partially harmonic with characteristic formants that remain constant for pitch shifts but change for different vowels. There are smaller inharmonic or noise like components for vowels, while the consonants are mostly of indefinite pitch but include strong resonance effects. Spoken language contains pitch shifts as a very characteristic feature. Attacks and decays are soft for both volume and timbre.

Subtractive emulation is hardly achievable: a strong formant filter may be realized by using either a lowpass with a medium resonance amount, a bandpass or a parametric filter (EQ). A better technique is to use an audio range oscillator with fixed frequency to modulate the frequency of a lowpass filter. The only problem: I know of no digital machine and not too many analogs that can do this. An upward pitch shift during attack and a downward pitch shift during release may help as well as some portamento. In general the vocal quality is very much dependent on medium rate pitch and timbre shifts during attack and release.

One FM method is to use a carrier with a fixed frequency and a ratio modulator (for the formant) combined with a harmonic spectrum from another stack. Another approach is to use a simple FM stack to create the formants but crossfade between operators with different frequency ratios to obtain the formant quality.

Expressive play is mandatory for convincing vocal emulations. Make the formant frequency resp. brightness very sensitive to key velocity and/or other controllers. Include volume swells, formant shifts and expressive vibrato in your phrasing. A breath controller is ideal here.

8.4 Fat "Analog" Sounds

If you are interested in typical "analog" synth sounds and maybe have a basic knowledge of analog synth programming, here are some starting points.

There are 2 algorithms which are nice for emulating a 2-oscillator or 3-oscillator synth:

- Algorithm #34 has two 3-stacks of operators.
- Algorithm #42 has three 2-stacks of operators.

For both cases set up feedback loops on the "highest" modulators, i.e. on operators 3 and 6 for algorithm #34 - see figure 9 - and on operators 2, 4 and 6 for algorithm #42 - see figure 10.

If all the frequencies in such a 2- or 3-stack are set to 1.0, the result is a saw-like spectrum. You can experiment with the levels and the feedback strength, usually set all carriers (i.e. 1, 4 or 1, 3, 5 for the two algorithms, respectively) to the highest level (127), the modulators a bit lower (80-110) and the feedback to 7 depending on the brightness/aliasing distortion you prefer. It's very nice to experiment with velocity sensitivity and realtime amplitude modulation for the modulators. This gives you much more expressiveness of play than in any sample player.

For just 2 stacked operators without feedback, a 1/1 frequency ratio of a carrier-modulator pair gives you a more saw-like spectrum (containing all harmonics), while a 1/2 ratio gives a square-like spectrum (only odd harmonics). Higher ratios result in pulse-like spectra (many harmonics missing, this sounds more "digital").

For envelopes, obviously the carrier envelopes control the loudness shape, while the modulator envelopes give something comparable to lowpass filter cutoff control in analog synths. It is very easy to program a "wah"-like attack timbre for instance. By choosing several modulators routed into one carrier and assigning envelopes with "shifted" peaks for the modulators, complex evolving timbres are possible.

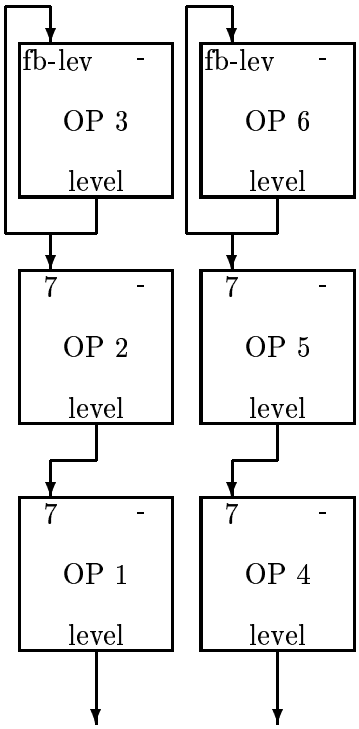

Figure 9: The "2 oscillator" algorithm #34 shown with feedback on the top modulators.

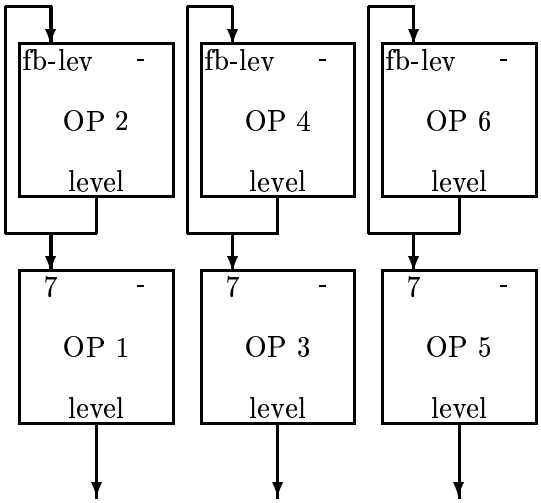

Figure 10: The "3 oscillator" algorithm #42 shown with feedback on the modulators.

To get an ultra fat synth sound, you can use a lot of techniques:

- Try detuning the operators/elements, especially by setting random tune to 1 (everything more is rather extreme).
- Set phase sync = off for some or all of the operators, this will result in a slightly differing sound for every key hit if there is any detuning. The effect is most noticeable with fast attack envelopes.
- Use a slight not too fast vibrato with a different amount on each of the operators. Use the main LFO with different PMS values. Also try the random waveform of the LFO.
- You can use operators as additional LFOs by setting a modulator to a low fixed frequency (see section 5.8 More LFOs).
- Individual timbre/volume LFOing can be obtained by looping the second half of the envelopes (e.g. loop point=2) with low rates and not too drastic level differences (even differences of 1 will do).
- Use the Pitch EG for a detune effect: Set all the rates to 0 and the levels to e.g. 0, -1, 2, -1, 1 and apply the Pitch EG to only one of the FM-stacks. The resulting sound is similar to drifting oscillators like instable analog VCOs. Don't use too many other detuning effects with this one.
- Of course you can use the FX section with some chorus or phasing, but don't overdo it. The nice thing about the FX section is that the LFOs in the modulation algorithms are not synchronized to the key ons, which results in random timbral variations especially for algorithms with very low modulation frequency. Check out using two or more modulation algorithms exclusively for the single elements in a voice.
- One nice thing is controlling the feedback level by velocity or realtime modulation using an "RCM loop": Select a 1 AFM & 1 AWM voice and set AWM wave = AFM and AFM external in = AWM. This gives you the output of the complete six operator AFM configuration routed through the AWM element as an additional feedback into any of your AFM operators. By using volume or filter control on the AWM element, you can achieve realtime control of the feedback level.

In fact all those techniques tend to make the sound diffuse or produce a "choral-tone", which is one possible meaning of "fat". Another criterion for fatness is a fast envelope setting that makes a sound punchy. The envelopes in the AFM section are very fast, so experiment with high attack rates for the modulators. If you want instant attack, set the starting level (L0) to 63. This way you might even get the typical click effect of a suddenly enabled oscillator: just set the corresponding oscillator phase to a non-zero value of the waveform (32 for sine) or set the oscillator sync to off.

For analog synth sounds, the most important control parameter is filter cutoff frequency and/or modulator level (brightness). The famous resonant filter sweep cannot really be simulated by FM, you have to use the filters either with a slow evolving envelope in mono mode or directly by assigning a MIDI controller (modulation wheel 2) to the filter cutoff.

9 SY Implementation

9.1 AWM

Looped samples are always played back from sample start and looped between loop start and loop end. The part between loop end and sample end is never played. Samples are transposed for less than $\pm$ 4 octaves. A bug in the sample readout algorithm will scan through the complete sample memory for very short loops.

minimum sample resolution (SDS)	8 bit
maximum sample resolution (SDS)	28 bit
minimum sample rate (SDS)	$\leq$ 5000 Hz
maximum sample rate (SDS)	$\geq$ 65535 Hz

9.2 AFM

The AFM chip contains 6 operators, 3 registers for intermediate storage (usually employed for feedback loops) plus one accumulator to sum up values (e.g. several modulators for one carrier). The operator output values are calculated from 6 to 1. The sampling rate seems to be exactly the same as on the DX7, i.e. a little below 60 000 Hz.

Here is the implementation of a simple FM-able oscillator in pseudo C, note that the modulator inputs do affect the momentary phase (phase register), but not the frequency itself:

```
double FM_oscillator( double frequency,     /* in Hertz */
                      double sampling_rate, /* in Hertz */
                      double FM_in1,
                      double FM_in2 )
{
static double phase;
double        phase_increment;

phase += frequency/sampling_rate;
phase %= 2*M_PI;

return( sin( 2*M_PI*( phase + FM_in1 + FM_in2 ) ) );
}
```

Here are the advantages of the AFM implementation in comparison to the DX7:

- fixed frequency goes down to 0 Hz (this enables waveshaping)
- initialization phase parameter, including free run
- 16 waveforms
- controllable operator input gains
- amplitude modulation by velocity can be negative
- individual realtime amplitude modulation (only positive)

- individual pitch modulation by main LFO (only positive) plus global pitch modulation by sub LFO
- individual on/off switch for pitch envelope
- better envelopes
- noise generator as independent source
- AWM output (sample playback) as independent source
- much more flexible routing (more algorithms, more connections possible, limited onboard algorithm editing, free algorithm editing via MIDI)

9.3 DX to SY Conversion

My assumption - backed up by several experiments - is that the 77 and 99 use a very similar FM implementation and the DX7 output level ranging from 0 to 99 is just a mapping into the more direct 0 to 127 range on the SYs. The mapping below is simply the "TL" table given in Chowning/Bristow [9]. The mapping is linear for high output levels which makes a lot of sense since the modulation index changes fast in that region. This mapping is applicable for the DX1, DX5, DX7, DX7s DX7II, TX7, TF1 (TX216, TX816), TX802 and DX9.

DX	SY	DX	SY	DX	SY	DX	SY	DX	SY	DX	SY	DX	SY	DX	SY	DX	SY	DX	SY
0	0	10	31	20	48	30	58	40	68	50	78	60	88	70	98	80	108	90	118
1	5	11	33	21	49	31	59	41	69	51	79	61	89	71	99	81	109	91	119
2	9	12	35	22	50	32	60	42	70	52	80	62	90	72	100	82	110	92	120
3	13	13	37	23	51	33	61	43	71	53	81	63	91	73	101	83	111	93	121
4	17	14	39	24	52	34	62	44	72	54	82	64	92	74	102	84	112	94	122
5	20	15	41	25	53	35	63	45	73	55	83	65	93	75	103	85	113	95	123
6	23	16	42	26	54	36	64	46	74	56	84	66	94	76	104	86	114	96	124
7	25	17	43	27	55	37	65	47	75	57	85	67	95	77	105	87	115	97	125
8	27	18	45	28	56	38	66	48	76	58	86	68	96	78	106	88	116	98	126
9	29	19	46	29	57	39	67	49	77	59	87	69	97	79	107	89	117	99	127

9.4 Modulation Index

The following table lists the modulation index as function of the operator output level for the DX7. This one is taken from Chowning/Bristows "FM Theory and Applications" [9].

DX7 Level	Modulation Index	DX7 Level	Modulation Index
99	13.119	60	0.446
95	9.263	50	0.188
90	6.029	40	0.079
85	3.894	30	0.031
80	2.512	20	0.013
75	1.639	10	0.003
70	1.068	0	0.000
65	0.690		

Chowning/Bristow also give a formula by which one can calculate the modulation index for all level settings. The slightly different numbers one gets from this formula probably result from rounding errors of π.

$$I = \pi \; 2^{33/16 - (127 - l)/8}$$

Below is a table that lists the modulation index as function of the output level parameter for the SY AFM section following the above formula. This one is based on the assumption that the SY series AFM implements the operator output level just as given in Chowning/Bristows "TL" table.

	0	1	2	3	4	5	6	7	8	9
0	$2.2{\cdot}10^{-4}$	$2.4{\cdot}10^{-4}$	$2.6{\cdot}10^{-4}$	$2.8{\cdot}10^{-4}$	$3.1{\cdot}10^{-4}$	$3.4{\cdot}10^{-4}$	$3.7{\cdot}10^{-4}$	$4.0{\cdot}10^{-4}$	$4.4{\cdot}10^{-4}$	$4.8{\cdot}10^{-4}$
10	$5.2{\cdot}10^{-4}$	$5.7{\cdot}10^{-4}$	$6.2{\cdot}10^{-4}$	$6.7{\cdot}10^{-4}$	$7.3{\cdot}10^{-4}$	$8.0{\cdot}10^{-4}$	$8.7{\cdot}10^{-4}$	$9.5{\cdot}10^{-4}$	$1.0{\cdot}10^{-3}$	$1.1{\cdot}10^{-3}$
20	$1.2{\cdot}10^{-3}$	$1.3{\cdot}10^{-3}$	$1.5{\cdot}10^{-3}$	$1.6{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$1.9{\cdot}10^{-3}$	$2.1{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$	$2.5{\cdot}10^{-3}$	$2.7{\cdot}10^{-3}$
30	$2.9{\cdot}10^{-3}$	$3.2{\cdot}10^{-3}$	$3.5{\cdot}10^{-3}$	$3.8{\cdot}10^{-3}$	$4.2{\cdot}10^{-3}$	$4.5{\cdot}10^{-3}$	$4.9{\cdot}10^{-3}$	$5.4{\cdot}10^{-3}$	$5.9{\cdot}10^{-3}$	$6.4{\cdot}10^{-3}$
40	$7.0{\cdot}10^{-3}$	$7.6{\cdot}10^{-3}$	$8.3{\cdot}10^{-3}$	$9.1{\cdot}10^{-3}$	$9.9{\cdot}10^{-3}$	0.0108	0.0118	0.0128	0.0140	0.0152
50	0.0166	0.0181	0.0198	0.0216	0.0235	0.0256	0.0280	0.0305	0.0332	0.0362
60	0.0395	0.0431	0.0470	0.0513	0.0559	0.0610	0.0665	0.0725	0.0791	0.0862
70	0.0940	0.1025	0.1118	0.1219	0.1330	0.1450	0.1581	0.1724	0.1880	0.2050
80	0.2236	0.2438	0.2659	0.2900	0.3162	0.3448	0.3760	0.4101	0.4472	0.4877
90	0.5318	0.5799	0.6324	0.6897	0.7521	0.8202	0.8944	0.9754	1.064	1.160
100	1.265	1.379	1.504	1.640	1.789	1.951	2.127	2.320	2.530	2.759
110	3.008	3.281	3.578	3.901	4.255	4.640	5.060	5.517	6.017	6.561
120	7.155	7.803	8.509	9.279	10.12	11.03	12.03	13.12		

9.5 Bessel Function Table

The following table lists the value of all Bessel functions against the operator output level setting l. Values smaller than $1 \cdot 10^{-4}$ are omitted or listed as 0. Note that all Bessel function are monotonic up to an output level of 104.

l	J_0	J_1
0	1.0	$1.09 \cdot 10^{-4}$
1	1.0	$1.19 \cdot 10^{-4}$
2	1.0	$1.30 \cdot 10^{-4}$
3	1.0	$1.42 \cdot 10^{-4}$
4	1.0	$1.54 \cdot 10^{-4}$
5	1.0	$1.68 \cdot 10^{-4}$
6	1.0	$1.84 \cdot 10^{-4}$
7	1.0	$2.00 \cdot 10^{-4}$
8	1.0	$2.18 \cdot 10^{-4}$
9	1.0	$2.38 \cdot 10^{-4}$
10	1.0	$2.60 \cdot 10^{-4}$
11	1.0	$2.83 \cdot 10^{-4}$
12	1.0	$3.09 \cdot 10^{-4}$
13	1.0	$3.37 \cdot 10^{-4}$
14	1.0	$3.67 \cdot 10^{-4}$
15	1.0	$4.00 \cdot 10^{-4}$
16	1.0	$4.37 \cdot 10^{-4}$
17	1.0	$4.76 \cdot 10^{-4}$
18	1.0	$5.19 \cdot 10^{-4}$
19	1.0	$5.66 \cdot 10^{-4}$
20	1.0	$6.18 \cdot 10^{-4}$
21	1.0	$6.74 \cdot 10^{-4}$
22	1.0	$7.34 \cdot 10^{-4}$
23	1.0	$8.01 \cdot 10^{-4}$
24	1.0	$8.73 \cdot 10^{-4}$
25	1.0	$9.52 \cdot 10^{-4}$
26	1.0	$1.04 \cdot 10^{-3}$
27	1.0	$1.13 \cdot 10^{-3}$

l	J_0	J_1
28	1.0	$1.24 \cdot 10^{-3}$
29	1.0	$1.35 \cdot 10^{-3}$
30	1.0	$1.47 \cdot 10^{-3}$
31	1.0	$1.60 \cdot 10^{-3}$
32	1.0	$1.75 \cdot 10^{-3}$
33	1.0	$1.90 \cdot 10^{-3}$
34	1.0	$2.08 \cdot 10^{-3}$
35	1.0	$2.27 \cdot 10^{-3}$
36	1.0	$2.47 \cdot 10^{-3}$
37	1.0	$2.69 \cdot 10^{-3}$
38	1.0	$2.94 \cdot 10^{-3}$
39	1.0	$3.20 \cdot 10^{-3}$
40	1.0	$3.49 \cdot 10^{-3}$
41	1.0	$3.81 \cdot 10^{-3}$
42	1.0	$4.15 \cdot 10^{-3}$
43	1.0	$4.53 \cdot 10^{-3}$
44	1.0	$4.94 \cdot 10^{-3}$
45	1.0	$5.39 \cdot 10^{-3}$
46	1.0	$5.88 \cdot 10^{-3}$
47	1.0	$6.41 \cdot 10^{-3}$
48	1.0	$6.99 \cdot 10^{-3}$
49	.9999	$7.62 \cdot 10^{-3}$
50	.9999	$8.31 \cdot 10^{-3}$
51	.9999	$9.06 \cdot 10^{-3}$
52	.9999	$9.88 \cdot 10^{-3}$
53	.9999	.0108
54	.9999	.0118
55	.9998	.0128

l	J_0	J_1	J_2	J_3
56	.9998	.0140	0	0
57	.9998	.0152	$1.16 \cdot 10^{-4}$	0
58	.9997	.0166	$1.38 \cdot 10^{-4}$	0
59	.9997	.0181	$1.64 \cdot 10^{-4}$	0
60	.9996	.0198	$1.95 \cdot 10^{-4}$	0
61	.9995	.0215	$2.32 \cdot 10^{-4}$	0
62	.9994	.0235	$2.76 \cdot 10^{-4}$	0
63	.9993	.0256	$3.28 \cdot 10^{-4}$	0
64	.9992	.0279	$3.91 \cdot 10^{-4}$	0
65	.9991	.0305	$4.64 \cdot 10^{-4}$	0
66	.9989	.0332	$5.52 \cdot 10^{-4}$	0
67	.9987	.0362	$6.57 \cdot 10^{-4}$	0
68	.9984	.0395	$7.81 \cdot 10^{-4}$	0
69	.9981	.0431	$9.28 \cdot 10^{-4}$	0
70	.9978	.0470	$1.10 \cdot 10^{-3}$	0
71	.9974	.0512	$1.31 \cdot 10^{-3}$	0
72	.9969	.0558	$1.56 \cdot 10^{-3}$	0
73	.9963	.0608	$1.86 \cdot 10^{-3}$	0
74	.9956	.0663	$2.21 \cdot 10^{-3}$	0
75	.9948	.0723	$2.62 \cdot 10^{-3}$	0
76	.9938	.0788	$3.12 \cdot 10^{-3}$	0
77	.9926	.0859	$3.71 \cdot 10^{-3}$	$1.07 \cdot 10^{-4}$
78	.9912	.0936	$4.41 \cdot 10^{-3}$	$1.38 \cdot 10^{-4}$
79	.9895	.1020	$5.24 \cdot 10^{-3}$	$1.79 \cdot 10^{-4}$
80	.9875	.1111	$6.22 \cdot 10^{-3}$	$2.32 \cdot 10^{-4}$
81	.9852	.1210	$7.40 \cdot 10^{-3}$	$3.01 \cdot 10^{-4}$
82	.9824	.1318	$8.79 \cdot 10^{-3}$	$3.90 \cdot 10^{-4}$
83	.9791	.1435	.0104	$5.05 \cdot 10^{-4}$

l	J_0	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
84	.9752	.1561	.0124	$6.55 \cdot 10^{-4}$	0	0	0	0	0
85	.9705	.1699	.0147	$8.48 \cdot 10^{-4}$	0	0	0	0	0
86	.9650	.1847	.0175	$1.10 \cdot 10^{-3}$	0	0	0	0	0
87	.9584	.2008	.0207	$1.42 \cdot 10^{-3}$	0	0	0	0	0
88	.9506	.2181	.0246	$1.84 \cdot 10^{-3}$	$1.03 \cdot 10^{-4}$	0	0	0	0
89	.9414	.2367	.0291	$2.38 \cdot 10^{-3}$	$1.46 \cdot 10^{-4}$	0	0	0	0
90	.9305	.2566	.0345	$3.08 \cdot 10^{-3}$	$2.05 \cdot 10^{-4}$	0	0	0	0
91	.9177	.2780	.0409	$3.98 \cdot 10^{-3}$	$2.90 \cdot 10^{-4}$	0	0	0	0
92	.9025	.3007	.0484	$5.14 \cdot 10^{-3}$	$4.08 \cdot 10^{-4}$	0	0	0	0
93	.8846	.3247	.0571	$6.63 \cdot 10^{-3}$	$5.75 \cdot 10^{-4}$	0	0	0	0
94	.8635	.3501	.0674	$8.55 \cdot 10^{-3}$	$8.10 \cdot 10^{-4}$	0	0	0	0
95	.8388	.3766	.0795	.0110	$1.14 \cdot 10^{-3}$	0	0	0	0
96	.8098	.4039	.0935	.0142	$1.60 \cdot 10^{-3}$	$1.44 \cdot 10^{-4}$	0	0	0
97	.7759	.4319	.1098	.0182	$2.25 \cdot 10^{-3}$	$2.21 \cdot 10^{-4}$	0	0	0
98	.7366	.4601	.1285	.0233	$3.15 \cdot 10^{-3}$	$3.38 \cdot 10^{-4}$	0	0	0
99	.6909	.4877	.1501	.0299	$4.41 \cdot 10^{-3}$	$5.17 \cdot 10^{-4}$	0	0	0
100	.6383	.5141	.1746	.0381	$6.15 \cdot 10^{-3}$	$7.89 \cdot 10^{-4}$	0	0	0
101	.5780	.5382	.2023	.0485	$8.56 \cdot 10^{-3}$	$1.20 \cdot 10^{-3}$	$1.40 \cdot 10^{-4}$	0	0
102	.5095	.5585	.2331	.0614	.0119	$1.82 \cdot 10^{-3}$	$2.32 \cdot 10^{-4}$	0	0
103	.4323	.5736	.2670	.0775	.0165	$2.76 \cdot 10^{-3}$	$3.84 \cdot 10^{-4}$	0	0
104	.3465	.5813	.3034	.0972	.0227	$4.17 \cdot 10^{-3}$	$6.34 \cdot 10^{-4}$	0	0
105	.2524	.5794	.3417	.1212	.0311	$6.27 \cdot 10^{-3}$	$1.04 \cdot 10^{-3}$	$1.48 \cdot 10^{-4}$	0
106	.1512	.5653	.3803	.1499	.0423	$9.37 \cdot 10^{-3}$	$1.71 \cdot 10^{-3}$	$2.65 \cdot 10^{-4}$	0
107	.0449	.5363	.4174	.1835	.0573	.0139	$2.78 \cdot 10^{-3}$	$4.73 \cdot 10^{-4}$	0
108	−0.0631	.4896	.4501	.2221	.0767	.0206	$4.51 \cdot 10^{-3}$	$8.40 \cdot 10^{-4}$	$1.36 \cdot 10^{-4}$
109	−0.1678	.4232	.4746	.2650	.1018	.0301	$7.25 \cdot 10^{-3}$	$1.48 \cdot 10^{-3}$	$2.62 \cdot 10^{-4}$
110	−0.2629	.3359	.4862	.3105	.1332	.0435	.0116	$2.59 \cdot 10^{-3}$	$5.04 \cdot 10^{-4}$
111	−0.3400	.2286	.4793	.3558	.1714	.0622	.0182	$4.50 \cdot 10^{-3}$	$9.60 \cdot 10^{-4}$
112	−0.3895	.1048	.4481	.3962	.2163	.0876	.0284	$7.72 \cdot 10^{-3}$	$1.81 \cdot 10^{-3}$
113	−0.4018	−0.0278	.3875	.4251	.2663	.1209	.0436	.0131	$3.38 \cdot 10^{-3}$

l	J_0	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
114	-0.3685	-0.1571	.2947	.4341	.3175	.1630	.0655	.0218	$6.21 \cdot 10^{-3}$
115	-0.2858	-0.2658	.1712	.4134	.3635	.2133	.0962	.0356	.0112
116	-0.1579	-0.3337	.0260	.3542	.3940	.2688	.1373	.0568	.0199
117	$-9.02 \cdot 10^{-4}$	-0.3404	-0.1225	.2516	.3961	.3227	.1888	.0879	.0343
118	.1553	-0.2733	-0.2461	.1097	.3555	.3630	.2478	.1312	.0574
119	.2690	-0.1362	-0.3105	-0.0531	.2620	.3725	.3057	.1866	.0925
120	.2972	.0414	-0.2857	-0.2011	.1171	.3319	.3468	.2498	.1418
121	.2148	.2019	-0.1631	-0.2855	-0.0564	.2276	.3482	.3078	.2041
122	.0395	.2732	.0247	-0.2616	-0.2092	.0649	.2855	.3377	.2701
123	-0.1534	.2041	.1974	-0.1190	-0.2744	-0.1176	.1477	.3086	.3178
124	-0.2493	.0136	.2520	.0860	-0.2010	-0.2449	-0.0410	.1963	.3126
125	-0.1649	-0.1821	.1319	.2299	$-6.93 \cdot 10^{-3}$	-0.2349	-0.2060	.0109	.2199
126	.0552	-0.2211	-0.0919	.1905	.1869	-0.0663	-0.2420	-0.1750	.0384
127	.2139	-0.0439	-0.2206	-0.0233	.2100	.1513	-0.0946	-0.2379	-0.1591

l	J_9	J_{10}	J_{11}	J_{12}	J_{13}	J_{14}	J_{15}	J_{16}
111	$1.80 \cdot 10^{-4}$	0	0	0	0	0	0	0
112	$3.73 \cdot 10^{-4}$	0	0	0	0	0	0	0
113	$7.65 \cdot 10^{-4}$	$1.55 \cdot 10^{-4}$	0	0	0	0	0	0
114	$1.55 \cdot 10^{-3}$	$3.44 \cdot 10^{-4}$	0	0	0	0	0	0
115	$3.09 \cdot 10^{-3}$	$7.55 \cdot 10^{-4}$	$1.66 \cdot 10^{-4}$	0	0	0	0	0
116	$6.04 \cdot 10^{-3}$	$1.63 \cdot 10^{-3}$	$3.95 \cdot 10^{-4}$	0	0	0	0	0
117	.0116	$3.45 \cdot 10^{-3}$	$9.21 \cdot 10^{-4}$	$2.23 \cdot 10^{-4}$	0	0	0	0
118	.0216	$7.13 \cdot 10^{-3}$	$2.10 \cdot 10^{-3}$	$5.61 \cdot 10^{-4}$	$1.37 \cdot 10^{-4}$	0	0	0
119	.0389	.0143	$4.68 \cdot 10^{-3}$	$1.38 \cdot 10^{-3}$	$3.72 \cdot 10^{-4}$	0	0	0
120	.0674	.0277	.0101	$3.30 \cdot 10^{-3}$	$9.82 \cdot 10^{-4}$	$2.68 \cdot 10^{-4}$	0	0
121	.1108	.0514	.0209	$7.62 \cdot 10^{-3}$	$2.52 \cdot 10^{-3}$	$7.60 \cdot 10^{-4}$	$2.12 \cdot 10^{-4}$	0
122	.1702	.0900	.0413	.0169	$6.19 \cdot 10^{-3}$	$2.08 \cdot 10^{-3}$	$6.41 \cdot 10^{-4}$	$1.84 \cdot 10^{-4}$
123	.2395	.1467	.0768	.0353	.0145	$5.44 \cdot 10^{-3}$	$1.86 \cdot 10^{-3}$	$5.91 \cdot 10^{-4}$
124	.2980	.2175	.1318	.0691	.0322	.0135	$5.16 \cdot 10^{-3}$	$1.82 \cdot 10^{-3}$
125	.3078	.2823	.2038	.1240	.0659	.0313	.0135	$5.31 \cdot 10^{-3}$
126	.2260	.2998	.2722	.1978	.1223	.0665	.0325	.0144
127	.0438	.2193	.2904	.2675	.1989	.1265	.0711	.0360

l	J_{17}	J_{18}	J_{19}	J_{20}	J_{21}	J_{22}
124	$5.96 \cdot 10^{-4}$	$1.82 \cdot 10^{-4}$	0	0	0	0
125	$1.94 \cdot 10^{-3}$	$6.58 \cdot 10^{-4}$	$2.09 \cdot 10^{-4}$	0	0	0
126	$5.90 \cdot 10^{-3}$	$2.24 \cdot 10^{-3}$	$7.92 \cdot 10^{-4}$	$2.63 \cdot 10^{-4}$	0	0
127	.0166	$7.07 \cdot 10^{-3}$	$2.80 \cdot 10^{-3}$	$1.04 \cdot 10^{-3}$	$3.61 \cdot 10^{-4}$	$1.19 \cdot 10^{-4}$

9.6 Pitch Shifting via LFO and PEG

Settings to achieve musical intervals via LFO or PEG. For the LFOs, PMS=7 is assumed. Exact matches are marked with *, the other values are optimal - but sometimes not very good - approximations to the intervals in equal temperament.

Shift [semitones]	LFO Depth Parameter	PEG Level (PEG range) 1/2	1	2	8
+1	11	+21	+11	+5	+1
+2	22	+43	+21	+11	+3
+3	33	+63*	+32*	+16*	+4*
+4	43, 44	-	+43	+21	+5
+5	54	-	+53	+27	+7
+6	65	-	+63*	+32*	+8*
+7	76	-	-	+37	+9
+8	87	-	-	+43	+11
+9	97	-	-	+48*	+12*
+10	108*	-	-	+53	+13
+11	119	-	-	+59	+15
+12	127	-	-	+63*	+16*
-1	11				
-2	22				
-3	32, 33	−64*	−32*	−16*	−4*
-4	43	-			
-5	54	-			
-6	64, 65	-	−64*	−32*	−8*
-7	75	-	-		
-8	86	-	-		
-9	96, 97	-	-		
-10	107	-	-		
-11	118	-	-		
-12	127	-	-		

9.7 LFO Frequency

The following frequencies were measured with varying precision.

Another note: don't assume anything about the phase of the LFO waveforms until you tested. The triangle wave of the main LFO starts with the maximum for example. I find it absolutely annoying that Yamaha implemented such nice features as LFO phase control and then made the usage completely counterintuitive.

LFO setting	frequency [Hz]	
	Main LFO	Sub LFO
0	1/32.5	0?
1	1/16.5	1/174
2	1/6.5	1/87
3	1/4.65	1/57
4	1/3.25	1/43.5
5	1/2.7	1/34.5
6	1/2.15	1/29
7	1/1.8	1/24.5
8	1/1.6	1/21.5
9	1/1.4	1/19.5
10	1/1.3	1/17.5
11	1/1.15	1/15.5
12	0.93	1/14.5
13	$\simeq$1	1/13.5
14	>1	1/12
25	2	
37	3	
43		1
51	4	
54		2
63	5	
64		3
66	6	
70	8	4
75	10	5
78	12	
80		6
84	15	
86		8
91		10
92	20	
96	22	12
97	23	
98	23.7	15
99	24.4	

Sub LFO setting	frequency [Hz]
100	
101	
102	20
103	22
104	23.5
105	25
106	30
107	32
108	35
109	38
110	41
111	44
112	47
113	53
114	59
115	65
116	70
117	75
118	80
119	88
120	94
121	105
122	116
123	130
124	140
125	153
126	164
127	175

9.8 Operator Envelopes

The operator envelopes are fast and exponential - which is very nice for most cases - but are very nonlinear in their behavior. With identical rate settings, a downward segment will be slower than an upward segment and with identical level parameter differences, a segment will be faster at high levels - even the output level influences the segement timing in this way! The graphic display does of course reflect nothing of this behavior, so look elsewhere and use your ears when synchonizing envelopes.

Another quirk is that the otherwise nice delay parameter (called HT = hold time) functions on a nonintuitive scale.

```
SLP = 4 (no loop)
L0 -> L1 -> L2 -> L3 -> L4 -> RL1 -> RL2
   R1    R2    R3    R4    RR1    RR2

SLP = 3
L0 -> L1 -> L2 -> L3 -> L4 -> L3 ... -> RL1 -> RL2
   R1    R2    R3    R4    R3        RR1    RR2

SLP = 2
L0 -> L1 -> L2 -> L3 -> L4 -> L2 -> L3 -> L4 ... -> RL1 -> RL2
   R1    R2    R3    R4    R2    R3    R4        RR1    RR2

SLP = 1
L0 -> L1 -> L2 -> L3 -> L4 -> L1 -> L2 -> L3 -> L4 ... -> RL1 -> RL2
   R1    R2    R3    R4    R1    R2    R3    R4        RR1    RR2
```

The following table contains rate settings for equal time upward and downward movements for level changes between 0 and 63 at full output level:

Rate up	Rate down
47	56 (!)
46	55 (!)
45	53 (!)
40	50 !
39	49
38	49
37	48
36	46
35	45
33	44 !
30	41
29	40 !
20	30

A Lesser known Facts

Here is just a list of miscellaneous things that may be helpful for sound-programming.

- The "volume low limit" parameter on the controller pages allows you to define another controller for volume - additional to MIDI's standard controller for volume #7. Both will be used in parallel, i.e. the last value sent by one of them is valid. This may result in volume jumps if you actually use both controllers. A volume low limit setting of zero will give full effect for the assigned controller.
- Pressing "Shift-EF Bypass" will assign the output sliders to the dry and wet signal level instead of output level 1 and 2. This function is indicated by a blinking effects LED.
- The SY99 does read single samples from disk in both TX16W and SY sample format. Both types of files need to have the extension `.Wnn` where `nn` is any integer between 01 and 99. The SY99 does write single samples always in SY sample format. Both the SY99 internal sample memory and the SY sample file format are organized 16-bit wide, so that storing a TX16W sample to disk will increase the file size and loading a sample with few quantization steps will not reduce its memory requirements.
- An internal test mode can be invoked by pressing "Voice", bank "D" and voice button "8" simultaneously. But be careful with the disk test, it may erase your floppy. An additional test mode can be invoked by pressing the number buttons "0" or "1" from normal test mode. Be sure to turn down the volume first, this test outputs a full level sine wave.
- The SYs MIDI clock signal can be shut off by setting "Sync" on the song main page to "MIDI" instead of "internal".
- The SY77, TG77 and SY99 voice dumps are partially compatible. The SY99 will receive a 77 voice dump but ignore the FX/output setting part. I assume it also works the other way round since the SY99 sends its voice dump in two parts, one of which contains only the parameters the SY77 and TG77 offer too.

B MIDI Standard

B.1 MIDI Note Numbers

octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

B.2 MIDI Controllers

B.2.1 14-Bit Controllers (MSB/LSB)

decimal	hex	function	remarks
00/32	00/20	Bank Select	
01/33	01/21	Modulation	
02/34	02/22	Breath	
03/35	03/23	(undefined)	formerly: Aftertouch
04/36	04/24	Foot Pedal	
05/37	05/25	Portamento Time	
06/38	06/26	Data Entry	
07/39	07/27	Main Volume	
08/40	08/28	Balance	
09/41	09/29	(undefined)	
10/42	0A/2A	Pan	
11/43	0B/2B	Expression	
12/44	0C/2C	Effect Control 1	
13/45	0D/2D	Effect Control 2	
14/46	0E/2E	(undefined)	
15/47	0F/2F	(undefined)	
16/48	10/30	General Purpose #1	formerly: Joystick X Axis
17/49	11/31	General Purpose #2	formerly: Joystick Y Axis
18/50	12/32	General Purpose #3	
19/51	13/33	General Purpose #4	
20/52-30/62	14/34-1E/3E	(undefined)	
31/63	1F/3F	(undefined)	formerly: Damp, Pitch Bend Sensitivity

B.2.2 7-Bit Controllers (formerly switches)

decimal	hex	function	remarks
64	40	Damper/Sustain/Hold #1	
65	41	Portamento On/Off	
66	42	Sostenuto	
67	43	Soft Pedal	
68	44	Legato Footswitch	
69	45	Hold #2	
70	46	Sound Controller #1	default: Sound Variation, formerly: Velocity Replace
71	47	Sound Controller #2	default: Timbre/Harmonic Intensity
72	48	Sound Controller #3	default: Release Time
73	49	Sound Controller #4	default: Attack Time
74	4A	Sound Controller #5	default: Brightness
75	4B	Sound Controller #6	default: undefined
76	4C	Sound Controller #7	default: undefined
77	4D	Sound Controller #8	default: undefined
78	4E	Sound Controller #9	default: undefined
79	4F	Sound Controller #10	default: undefined
80	50	General Purpose #5	
81	51	General Purpose #6	
82	52	General Purpose #7	
83	53	General Purpose #8	
84	54	Portamento Control	
85-90	55-5A	(undefined)	
91	5B	Effect 1 Depth	formerly: External Effects Depth
92	5C	Effect 2 Depth	formerly: Tremolo Depth
93	5D	Effect 3 Depth	formerly: Chorus Depth
94	5E	Effect 4 Depth	formerly: Celeste Depth
95	5F	Effect 5 Depth	formerly: Phaser Depth
96	60	Data Increment	
97	61	Data Decrement	
98	62	Non-Registered Parameter LSB	
99	63	Non-Registered Parameter MSB	
100	64	Registered Parameter LSB	
101	65	Registered Parameter MSB	
102	66	*Mono Pitch	proposed; pending MMA & JMSC
103-118	67-76	(undefined)	

B.2.3 Channel Mode Messages

decimal	hex	function	remarks
119	77	*Mute Channel	proposed; pending JMSC
120	78	All Sound Off	
121	79	Reset All Controllers	
122	7A	Local Control On/Off	
123	7B	All Notes Off	
124	7C	Omni Mode Off	
125	7D	Omni Mode On	
126	7E	Mono Mode Off	
127	7F	Mono Mode On	

B.2.4 Registered Parameters (MSB/LSB):

decimal	hex	function	remarks
00/00	00/00	Pitch Bend Sensitivity	
00/01	00/01	Fine Tuning	
00/02	00/02	Coarse Tuning	
00/03	00/03	Tuning Program Select	
00/04	00/04	Tuning Bank Select	
127/127	7F/7F	*Null Controller	proposed; pending MMA

C General MIDI

C.1 GM Patch Map

#	Instrument	#	Instrument	#	Instrument
1-8	**Piano**	9-16	**Chromatic Percussion**	17-24	**Organ**
1	Acoustic Grand	9	Celesta	17	Drawbar Organ
2	Bright Acoustic	10	Glockenspiel	18	Percussive Organ
3	Electric Grand	11	Music Box	19	Rock Organ
4	Honky-Tonk	12	Vibraphone	20	Church Organ
5	Electric Piano 1	13	Marimba	21	Reed Organ
6	Electric Piano 2	14	Xylophone	22	Accoridan
7	Harpsichord	15	Tubular Bells	23	Harmonica
8	Clav	16	Dulcimer	24	Tango Accordian

#	Instrument	#	Instrument	#	Instrument
25-32	**Guitar**	33-40	**Bass**	41-48	**Strings**
25	Acoustic Guitar (nylon)	33	Acoustic Bass	41	Violin
26	Acoustic Guitar (steel)	34	Electric Bass (finger)	42	Viola
27	Electric Guitar (jazz)	35	Electric Bass (pick)	43	Cello
28	Electric Guitar (clean)	36	Fretless Bass	44	Contrabass
29	Electric Guitar (muted)	37	Slap Bass 1	45	Tremolo Strings
30	Overdriven Guitar	38	Slap Bass 2	46	Pizzicato Strings
31	Distortion Guitar	39	Synth Bass 1	47	Orchestral Strings
32	Guitar Harmonics	40	Synth Bass 2	48	Timpani

#	Instrument	#	Instrument	#	Instrument
49-56	**Ensemble**	57-64	**Brass**	65-72	**Reed**
49	String Ensemble 1	57	Trumpet	65	Soprano Sax
50	String Ensemble 2	58	Trombone	66	Alto Sax
51	SynthStrings 1	59	Tuba	67	Tenor Sax
52	SynthStrings 2	60	Muted Trumpet	68	Baritone Sax
53	Choir Aahs	61	French Horn	69	Oboe
54	Voice Oohs	62	Brass Section	70	English Horn
55	Synth Voice	63	SynthBrass 1	71	Bassoon
56	Orchestra Hit	64	SynthBrass 2	72	Clarinet

#	Instrument	#	Instrument	#	Instrument
73-80	**Pipe**	81-88	**Synth Lead**	89-96	**Synth Pad**
73	Piccolo	81	Lead 1 (square)	89	Pad 1 (new age)
74	Flute	82	Lead 2 (sawtooth)	90	Pad 2 (warm)
75	Recorder	83	Lead 3 (calliope)	91	Pad 3 (polysynth)
76	Pan Flute	84	Lead 4 (chiff)	92	Pad 4 (choir)
77	Blown Bottle	85	Lead 5 (charang)	93	Pad 5 (bowed)
78	Skakuhachi	86	Lead 6 (voice)	94	Pad 6 (metallic)
79	Whistle	87	Lead 7 (fifths)	95	Pad 7 (halo)
80	Ocarina	88	Lead 8 (bass+lead)	96	Pad 8 (sweep)

#	Instrument	#	Instrument	#	Instrument
97-104	**Synth Effects**	105-112	**Ethnic**	113-120	**Percussive**
97	FX 1 (rain)	105	Sitar	113	Tinkle Bell
98	FX 2 (soundtrack)	106	Banjo	114	Agogo
99	FX 3 (crystal)	107	Shamisen	115	Steel Drums
100	FX 4 (atmosphere)	108	Koto	116	Woodblock
101	FX 5 (brightness)	109	Kalimba	117	Taiko Drum
102	FX 6 (goblins)	110	Bagpipe	118	Melodic Tom
103	FX 7 (echoes)	111	Fiddle	119	Synth Drum
104	FX 8 (sci-fi)	112	Shanai	120	Reverse Cymbal

#	Instrument
121-128	**Sound Effects**
121	Guitar Fret Noise
122	Breath Noise
123	Seashore
124	Bird Tweet
125	Telephone Ring
126	Helicopter
127	Applause
128	Gunshot

C.2 GM Drum Map

Note #	Drum Sound
35	Acoustic Bass Drum
36	Bass Drum 1
37	Side Stick
38	Acoustic Snare
39	Hand Clap
40	Electric Snare
41	Low Floor Tom
42	Closed Hi-Hat
43	High Floor Tom
44	Pedal Hi-Hat
45	Low Tom
46	Open Hi-Hat
47	Low-Mid Tom
48	Hi-Mid Tom
49	Crash Cymbal 1
50	High Tom

Note #	Drum Sound
51	Ride Cymbal 1
52	Chinese Cymbal
53	Ride Bell
54	Tambourine
55	Splash Cymbal
56	Cowbell
57	Crash Cymbal 2
58	Vibraslap
59	Ride Cymbal 2
60	Hi Bongo
61	Low Bongo
62	Mute Hi Conga
63	Open Hi Conga
64	Low Conga
65	High Timbale
66	Low Timbale

Note #	Drum Sound
67	High Agogo
68	Low Agogo
69	Cabasa
70	Maracas
71	Short Whistle
72	Long Whistle
73	Short Guiro
74	Long Guiro
75	Claves
76	Hi Wood Block
77	Low Wood Block
78	Mute Cuica
79	Open Cuica
80	Mute Triangle
81	Open Triangle

D What is still missing

- RCM: AWM as a carrier
- RCM: AWM-carriers by free algorithm edit
- general sound character of FM, resonances
- high stacks: delay line
- FM drums
- drums by filtering etc
- use of the oscillator phase parameter, non-repetetive key-on
- types of beatings/animation techniques
- waveshaping theory
- feedback theory
- Bessel graphs
- example spectrum graphs
- envelope programming (loops, nonlinearity, mod intensity)
- LFO programming
- filter programming (EG vs. EG-VA)

- additive synthesis
- layering
- creative use of samples
- Synthesis Methods: VOSIM, Pulsformung, iPD
- good FM explanation
- simple FM spectrum rules
- discussion of FM waveforms
- Emulations: Piano, E-Piano, Harpsichord/D6, Sax, Flute, Clarinet, Electric Bass, Guitar, Drums, Bells, Synth Percussion, Special FX
- more and better references (more quotes, authors)
- good structure
- good English

And a lot more...

Index

References

[1] Craig Anderton. *The Digital Delay Handbook.* Amsco Publications, NY, 1985. ISBN: 0-8256-2414-2.

[2] James Beauchamp. Brass-tone synthesis by spectrum evolution metching with nonlinear functions. *Computer Music Journal*, 3, 2:35 – 43, 1979. reprinted in [28].

[3] Gerald Bennett and Xavier Rodet. Synthesis of the singing voice. In Max Matthews and John Pierce, editors, *Current Directions in Computer Music Research*, pages 19 – 44. Cambridge MA: MIT Press, 1989.

[4] Claudius Brüse. Physical modeling. In *Keyboards Magazin.* 4/1995.

[5] Richard Cann. An analysis/synthesis tutorial. In Curtis Roads and John Strawn, editors, *Foundations of computer music*, pages 114 – 144. Cambridge MA: MIT Press, 1985. originally appeared in Computer Music Journal 3, 3: 6-11, 1979; 3, 4: 9-13, 1979; 4,1: 36-42, 1980.

[6] Wendy Carlos. *The Secrets of Synthesis.* CBS MK 42333, 1987. CD with booklet.

[7] John M. Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21, 7:526 – 534, 1973. reprinted in [28].

[8] John M. Chowning. Frequency modulation synthesis of the singing voice. In Max Matthews and John Pierce, editors, *Current Directions in Computer Music Research*, pages 57 – 63. Cambridge MA: MIT Press, 1989.

[9] John M. Chowning and Dave Bristow. *FM Theory and Applications by Musicians for Musicians.* Yamaha Music Foundation, Tokyo, 1986. ISBN: 4-636-17482-8 (out of print).

[10] Steve de Furia. *The Secrets of Analog and Digital Synthesis.* Hal Leonard Books, 1985. ISBN: 0-88188-516-9.

[11] Peter Gorges. *Das komplette DX7 Handbuch.* GC Gunther Carstensen Verlag, München, 1988. ISBN: 0-9802026-0-7.

[12] Peter Gorges. *Das große Sampler Praxisbuch.* GC Gunther Carstensen Verlag, München, 1991. ISBN: 3-910092-00-2.

[13] Peter Gorges. Soundforum. In *Keyboards Magazin.* 5/1993-today.

[14] Peter Gorges and Alex Merck. *Keyboards, MIDI, Homerecording.* GC Gunther Carstensen Verlag, München, 1989. ISBN: 3-9802026-3-1.

[15] Hubert Henle. *Das Tonstudio Handbuch.* GC Gunther Carstensen Verlag, München, 1990. ISBN: 3-9802026-5-8.

[16] Julius O. Smith III. Digital waveguide modeling of musical instruments. *Computer Music Journal*, 16, 4:74 – 91, 1992. revised and extended 1993.

[17] Marc Lebrun. A derivation of the spectrum of FM with a complex modulating wave. *Computer Music Journal*, 1, 4:51 – 52, 1977. reprinted in [28].

[18] Howard Massey. *The Complete DX7II.* Amsco Publications, 1987. ISBN: 0-8256-1119-9 (out of print).

[19] Howard Massey. *SY77 Applications Guidebook.* Yamaha Corporation, 1989.

[20] Max Matthews and John Pierce, editors. *Current Directions in Computer Music Research.* Cambridge MA: MIT Press, 1989. ISBN: 0-262-13241-9 (hardcover), 0-262-63139-3 (paperback).

[21] F. Richard Moore. Table lookup noise for sinusoidal digital oscillators. *Computer Music Journal*, 1, 2:26 – 29, 1977. reprinted in [28].

[22] F. Richard Moore. *Elements of computer music.* Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN: 0-13-252552-6.

[23] Dexter Morrill. Trumpet algorithms for computer composition. In Curtis Roads and John Strawn, editors, *Foundations of computer music*, pages 30 – 44. Cambridge MA: MIT Press, 1985.

[24] John R. Pierce. Klang - Musik mit den Ohren der Physik. Spektrum Verlag, Heidelberg.

[25] Jeff Pressing. *Synthesizer performance and real-time techniques.* Madison, Wisconsins : A-R Editions, 1992. ISBN: 0-89579-257-5.

[26] Curtis Roads. Granular synthesis of sound. In Curtis Roads and John Strawn, editors, *Foundations of computer music*, pages 145 – 159. Cambridge MA: MIT Press, 1985. revised version of a paper that appeared in Computer Music Journal 2, 2: 61-62, 1978.

[27] Curtis Roads. A tutorial on nonlinear distortion or waveshaping synthesis. In Curtis Roads and John Strawn, editors, *Foundations of computer music*, pages 83 – 94. Cambridge MA: MIT Press, 1985. revised version of a paper that appeared in Computer Music Journal 3, 2: 29-34, 1979.

[28] Curtis Roads and John Strawn, editors. *Foundations of computer music.* Cambridge MA: MIT Press, 1987.

[29] Thomas D. Rossing. *The Science of Sound 2nd edition.* Addison Wesley, 1990. ISBN: 0201-15727-6.

[30] Steve Saunders. Improved FM audio synthesis methods for real-time digital music generation. *Computer Music Journal*, 1, 1:53 – 55, 1977. reprinted in [28].

[31] Andreas Schätzl. *Das Roland D-50/550 Praxisbuch.* Signum Verlag, München, 1988. ISBN: 3-924767-24-6.

[32] Bill Schottstaedt. The simulation of natural instrument tones using frequency modulation with a complex modulating wave. *Computer Music Journal*, 1, 4:46 – 50, 1977. reprinted in [28].

[33] several authors. Praxis. In *KEYS Magazin*. 1/1993-today.

[34] John Strawn, editor. *Digital audio signal processing*. Los Altos, CA: W. Kaufmann, 1985. ISBN: 0-86576-082-9.

[35] Barry Truax. Organizational techniques for c:m ratios in frequency modulation. *Computer Music Journal*, 1, 4:39 – 45, 1977. reprinted in [28].

[36] Wolfgang Wagner, Oliver Wetter, Martin Schneider, and Georg Näger. *Digitale Signalprozessoren für Audio*. Elektor Verlag, Aachen, 1994. ISBN: 3-928051-50-4.

[37] Bradley Wait, editor. *Guitar Synth and MIDI*. Hal Leonhard Books, 1988. ISBN: 0-88188-593-2.