Artist's Research Statement

Justin Yang
BESA in Music Technology and Electrical and Computer Engineering

Introduction

The Yamaha DX7 is the first commercially successful digital synthesizer. Originally released in 1983, the Yamaha DX7 quickly took over popular culture and appeared in virtually every single 80s pop music. Despite its success, Yamaha DX7 along with the frequency modulation (FM) algorithm it uses for sound generations are also known to be notoriously hard to understand and program. Hence, the original goal of my capstone project is to create a hardware programmer that can help musicians/users better understand and utilize the sound design capability of the DX7 and FM synthesis in general. However, as the project evolved over the year and as I learned more about FM synthesis, my capstone project became more about analyzing the different FM synthesizer interface design and understanding why it is so difficult to make an effective and intuitive interface for FM synthesizers.

Inception

The inception of my capstone project could be traced back to as early as Fall Semester of my Freshman Year when I first acquired my own DX7, specifically the DX7iiD. Being the first ever vintage synthesizer I own, I spent hours playing it, falling in love with the diversity of sounds it generates. However, as I also started trying to program my own sounds with the synthesizer, it quickly became a frustrating experience of having to navigate several layers of menu to adjust one single parameter and getting confused with the countless almost no-ending list of parameters it has. To make matters worse, when I tried to understand how FM synthesis generates harmonics, or timbre in musical terms, I was bombarded with mathematical equations and terms that I just didn't understand at the time. Eventually as college went on, learning how to program and utilizing DX7's sound design capabilities became more of a back burner project.

When we were tasked with figuring out a capstone project idea last year, my experiences with the DX7 as well as the long-time desire to learn how to program it came to my mind. I realized that having been in CMU for almost 4 years and having gone through ECE and Music Technology curriculum, the mathematical equations that used to make no sense to me now look like just another equation I now encounter on a daily basis. Additionally, since Freshman year, I have also encountered and used other FM synthesizers with drastically different programming interfaces. All of them got me thinking, is there a way I could utilize my ECE knowledge to make programming the DX7 or any FM synthesizers in general more intuitive and friendly to musicians/users who don't have a technical background? Additionally, what exact features of the DX7 that makes its programming interface so hard to use? At the same time, what were the intentions in designing DX7's interface the way it is factoring the limitations they had at the time? And what choices would I make to make my version more approachable? Lastly, going

back to my Freshman year's goal, could I finally understand how the synthesizer works and program my own sounds on it?

Photo of my DX7iiD when it first arrived at my Freshman Year dorm

Research

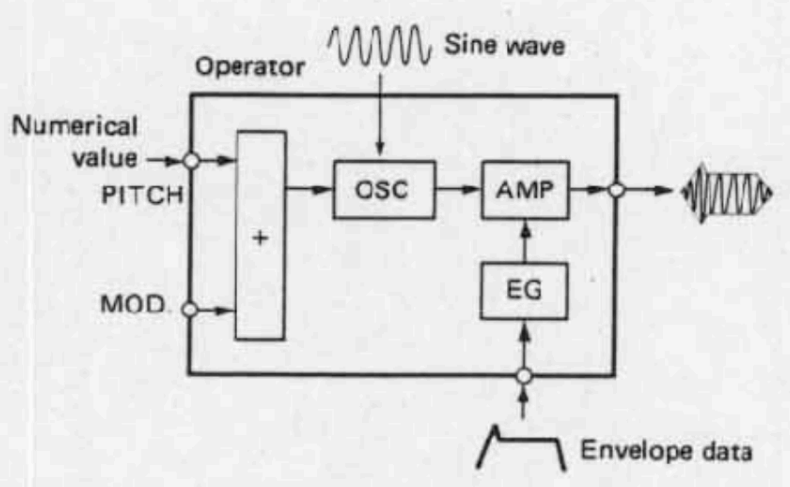
Basics of FM Synthesis

FM synthesis is a sound synthesis method that uses frequency modulation in its core to generate harmonics or timbre in musical terms. Frequency modulation in the simplest sense just means a change in frequency or pitch in musical terms. A good example of a real-life frequency modulation phenomenon is vibrato in singing or instruments. When vibrato happens in music, the pitch from the singer or the instrument shifts back and forth between a slightly higher and lower pitch creating a "pulsating effect". Since the pitch is swinging back and forth, this pitch change can be described as a wave and we can say this described wave is frequency modulating the sound that the singer and instrument produces, which keeps in mind is also a wave. The idea behind FM synthesis is the question of what happens if this vibrato or pitch change happens at an even faster rate (Vibrato is usually only about 5 - 8 Hz), so fast that it matches the frequency of the sound it changes? (For instance if the sound you play has a frequency of 440 Hz, the rate of the pitch change also is 440 Hz) The answer is that if the sound you start with happens to be a sine wave, which we perceive as a pure tone with no harmonics, modulating this sine wave at a fast rate would end up "magically" generating harmonics effectively out of nothing. What that means in music and sound sense is that FM synthesis can be easily used to generate a full spectrum of harmonics with effectively 2 pure tones.

In essence, the very basic FM synthesis algorithm will always involve 2 waves: carrier and modulator wave. The wave that we primarily hear and has its pitch changed/modulated is called the carrier. The wave that describes the pitch change and modulates the carrier is called the modulator. (It is important to note that whether a wave being a modulator or a carrier is relative in FM synthesis. As we will see later on, modulator waves themselves can be modulated by other modulators.) Additionally, for simplicity's sake, usually these 2 waves are assumed to be sine waves which are also the default waveforms for FM synthesizers like the DX7.

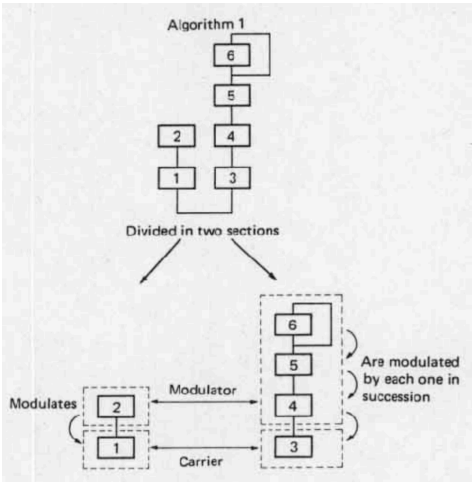
The last component to understand the basics of FM synthesis is the idea of an operator. As mentioned before, whether a wave is a carrier or modulator depends on the role it plays in the

FM synthesis algorithm or how it is routed. In simple terms, if we can hear it, the wave is a carrier. If we can't, the wave is a modulator. To allow the devices that generate these waves, which we call oscillators in synthesizer and electrical engineering term, to be able to be dynamically used as both the modulator and carrier, the original DX7 engineers decided to abstract these oscillators, have each of them tie to an envelope that controls their amplitudes, and together with the envelope form a unit called the operator.



(Block diagram of an operator)
(Source: *DX7 Operating Manual*, p9)

When an operator acts as a carrier, its envelope controls the carrier's volume/loudness over time. (Ex. envelope can make the sound rise up in volume for the first few milliseconds, and stay at the volume while the key is pressed, and later have it decays back to zero once the key is lifted.) When it acts as a modulator, the envelope will instead affect how the timbre/harmonics changes over time since the amplitude/level of the modulator is a key parameter in changing how the timbre/harmonics of FM synthesizers behave. Lastly, how the operators (6 operators for DX7) are arranged to be carriers, modulators, or even modulators modulating modulators is called an algorithm in FM synthesis terms.



(An example of how to interpret FM synthesizer's algorithm block diagram)
(Source: *DX7 Operating Manual*, p11)

Basic Equation of FM Synthesis

The basic equation of a carrier sine wave being frequency modulated by another sine wave is:

$$e = A \sin(\alpha t + I \sin\beta t) \qquad (1)$$

where

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

With the main parameters of frequency modulation being:

c = carrier frequency or average frequency
m = modulating frequency
d = peak deviation.

(Source: "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation" by John Chowning)

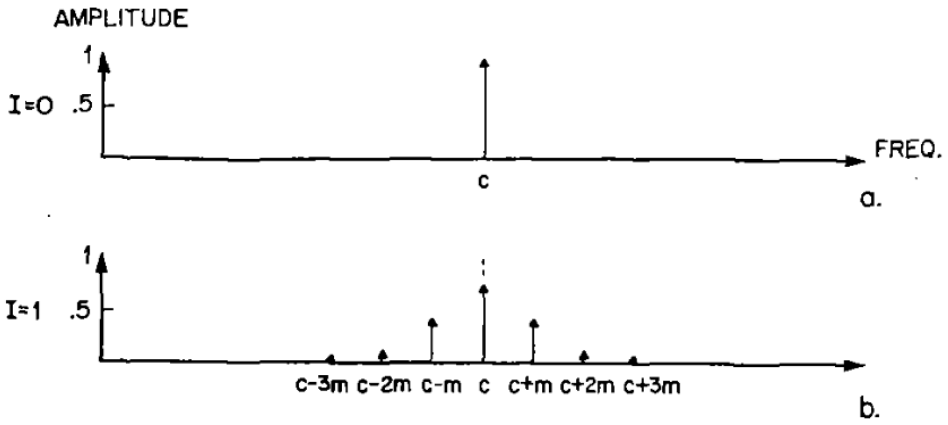
A couple of observations of the equation:

1. While modulation index I is mentioned to be a ratio or the peak deviation and modulating frequency, I is also just the amplitude of the modulating sine wave.
2. For $I = d/m$, even though the above equation doesn't fully tell us how FM synthesis behaves in the frequency domain, I and m's relationship to d does imply I and m play a role in determining the spectrum FM synthesis produces.
3. In order to have any frequency modulation and generation of harmonics, I has to be greater than 0.
4. Note usually in actual implementation only I and m can be directly changed.

While this equation tells us how the waveform generated by FM synthesis looks like, it doesn't directly tell us what the exact harmonics and their relative amplitudes the synthesis generate. Or in other words, this equation doesn't tell us how FM synthesis sounds in relation to its parameters. To predict how FM synthesis generates its harmonics, we need to use Bessel Functions and also understand how it generates "sidebands".

How FM synthesis generates sidebands/harmonics

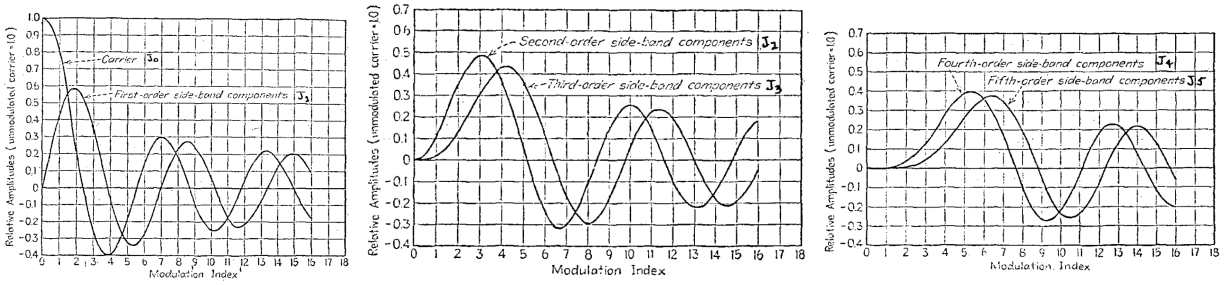
One quick observation we can make about FM synthesis is that when a carrier sine wave is being modulated (ie. modulation index is not 0), it tends to produce sidebands centered around the carrier sine wave's frequency.



(c in this example is the carrier wave's frequency and m is frequency of the modulator)
(Source: "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation" by John Chowning)

The exact frequencies of these sidebands it turns out can be calculated by the equation $\underline{c + km}$ where k is an integer that can be both negative and positive. There are technically infinite amounts of sidebands and in general k goes from negative infinity to positive infinity. But the more sidebands approach either end of the graph, the closer they get to zero in amplitude as a result of how Bessel Functions work. So in practice you will only really see non-zero sidebands appearing within the range of $-(I + 2)$ to $(I + 2)$ with I being the modulation index. I and k are also importantly used as the key parameters for determining the exact amplitude of the corresponding sideband through Bessel Functions.

Bessel Functions are often denoted as $J_n(X)$ where n is the order and X is the input of Bessel Functions. n in many ways represent the "nth Bessel Function" we are using as Bessel Functions can be thought of as a collection of functions indexed by n. To get the exact amplitude of sideband, we simply set $n = k$ and $X = I$. In other words, the amplitude of the kth sideband at the frequency $c + km$ with modulation index I is equal to $J_k(I)$.

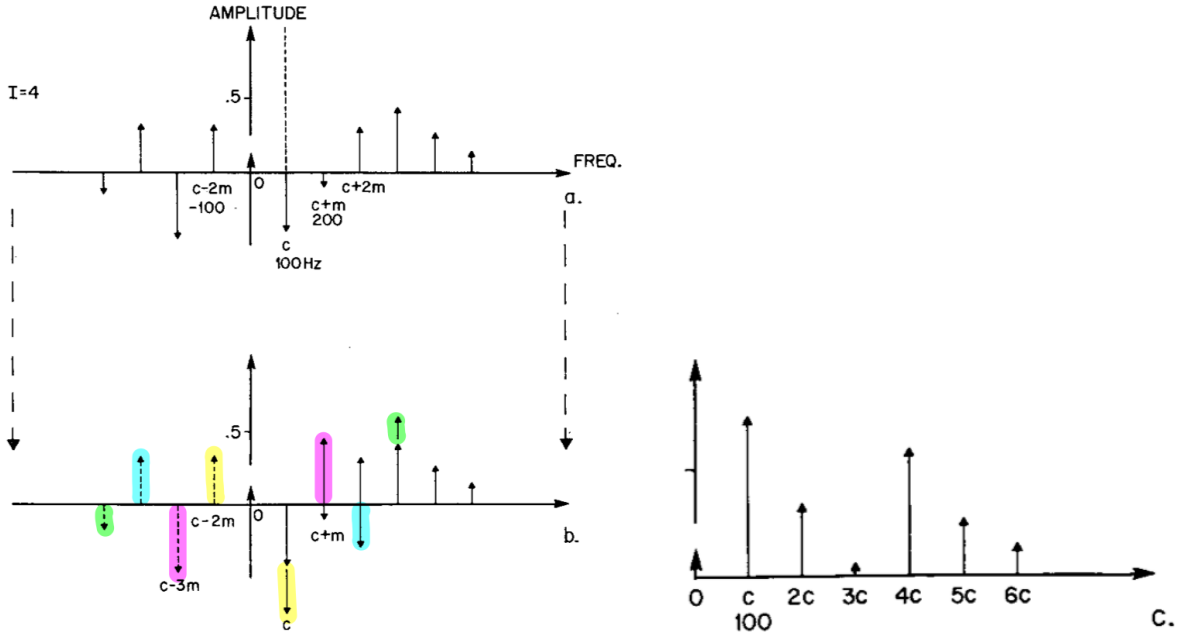


(Here are the first six Bessel Functions from J_0 - J_5. Observe how the Bessel Functions do in fact output negative values and zero for its amplitude for certain modulation indexes. This will be important later on.)
(Source: "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation" by John Chowning)

Another thing to mention is that as modulation index *I* increases, the sidebands spread further out. The next question is what happens when sidebands that go toward negative infinity exceed 0 Hz and go into the negative frequency?

What happens is that the sidebands that went into negative frequency are reflected around 0 Hz and "mix" with sidebands in the positive frequency. This happens because of Hermitian symmetry and the fact that the Fourier transform of a sine wave produces 2 "frequency bands" (delta functions to be more exact) with one at the positive frequency of sine wave and the other at the negative frequency of the sine wave. In other words, when the spectrum expands away from positive carrier frequency toward both the positive and negative frequency, we have the same process happening somewhat symmetrically on the negative carrier frequency band as well. Effectively speaking, the reason why the sidebands that go into negative frequency "reflected around the 0 Hz and mix with sidebands in the positive frequency" is because they came from sidebands that went over 0 Hz generated from the negative frequency component of the carrier sine wave.

To make matters even more complicated, the amplitudes for the 2 side bands of sine waves happen to have opposite amplitudes meaning the spectrum that "gets reflected" over will also need to be timed by -1 before being mixed into the rest of the positive sidebands. As a result, some positive side bands will be attenuated and some will be increased in this process.



(a. Spectrum calculated from Bessel Functions with some sidebands going into negative frequency. b. sidebands that are in the negative frequencies get reflected over but before they are added to the rest of the positive sidebands, their amplitudes get timed by -1. c. the final spectrum as perceived by humans)
(Source: "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation" by John Chowning)

Harmonic vs. Inharmonic Spectra

FM synthesis can generate both harmonic and inharmonic spectra. So far the examples used in this document are all harmonic spectra examples. Harmonic spectra happens when the ratio between the frequency of carrier *c* and modulator *m* is rational with $c/m = N_1/N_2$ where N_1 and N_2 are both natural numbers. The reason this produces a harmonic spectrum is because the sidebands that get reflected over are guaranteed to overlap with another positive sideband of the spectrum and as a result, the sidebands' frequencies will guarantee to still follow the harmonic series, resulting in a harmonic spectrum. On the other hand, when the ratio c/m is irrational or N_1 and N_2 are not natural numbers, the reflected sidebands will not always overlap with the existing positive sidebands. Instead some of the reflected over sidebands will be in between the existing positive sidebands causing the inharmonic sound.

DX7's level scaling and modulation index

Despite modulation index being one of the most important parameters for FM synthesis, surprisingly modulation index is not a parameter the DX7 has, rather modulation index is controlled/implied through operator output level when that specific operator is used as a modulator. The reason why Yamaha did this is because the operators can be configured as either modulators or carriers depending on the algorithm used. By not having the modulation index as a direct control, Yamaha allowed a design where all operators have the same identical control interface.

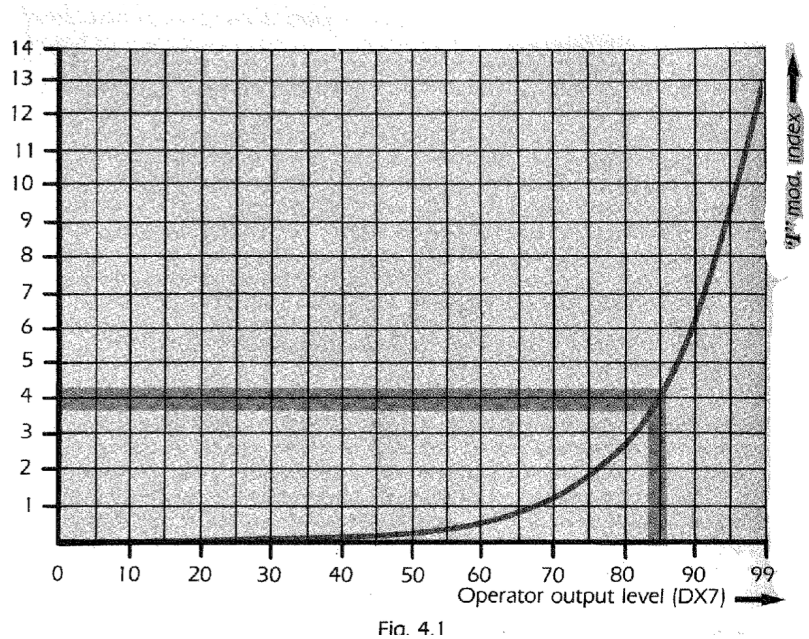


Fig. 4.1

(Graph of DX7's operator output level and their corresponding modulation index)

(Source: *FM Theory & Applications: By Musicians for Musicians* by David Bristow and John Chowning, p58)

Complex FM Synthesis

Besides just configuring FM synthesis to be just one operator acting as the modulator to the carrier operator, DX7 also offered other more complicated methods to arrange the operators

including: 1. one operator modulating several carriers, 2. several modulators summed and modulate one carrier, 3. cascade or “stacked” FM, and 4. feedback FM.

1. One operator modulating several carriers: Since the shared modulator is still very much outputting sinusoids, the math to understand how it generates harmonics is also very much identical to how normal FM generates harmonics. In practice, this is a very efficient way to allow the carriers in the algorithm to all be modulated without having to assign another operator to be the modulator, especially since DX7 has a limit of 6 operators. The only major downside is that since the carriers share the same operator, they also share the same envelope that controls how their harmonics evolve over time.

2. Several modulators summed and modulate one carrier: The outputs of the modulators are first summed up into a signal that modulates the carrier. Because any periodic signal can be expressed as a sum of sinusoids (Fourier Series), this form of complex FM is analogous to a normal FM but with a non-sinusoid modulator modulating a carrier instead of the typical sinusoid modulator. This process results in the following equation:

$$\sin(\omega_c t + B_1 \sin\omega_{m1} t + B_2 \sin\omega_{m2} t) = \sum_{i=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} J_i(B_1) J_k(B_2) \sin(\omega_c + i\omega_{m1} + k\omega_{m2})t$$

(Source: https://ccrma.stanford.edu/software/snd/snd/fm.html)

From the right hand side of the equation above, the double summation part as long as the sin term tell us there will be even more sidebands meaning that compared to normal FM, this method can result in a much “denser” spectrum

3. Cascade or “stacked” FM: Cascade or “stacked” FM happens when the modulator modulating the carrier is also a carrier being modulated by another operator, to which it results in the following equation:

$$\sin(\omega_c t + B_1 \sin(\omega_{m1} t + B_2 \sin(\omega_{m2} t))) = \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} J_n(B_1) J_k(nB_2) \sin(\omega_c + n\omega_{m1} + k\omega_{m2})t$$

(Source: https://ccrma.stanford.edu/software/snd/snd/fm.html)

The equation is surprisingly similar to the several modulator summed equations from above. The only difference is that B_2 being scaled by n in the $J_k(nB_2)$ compared to $J_k(B_2)$ in the equation above.

4. Feedback FM: Feedback FM happens when output of an operator goes toward the pitch input of the operator. Unlike other FM, this form of FM only requires 1 operator as in this case the same operator acts as both the carrier and modulator. It results in the following equation:

$$\sum_{n=1}^{\infty} \frac{2}{nB} J_n(nB) \sin n\omega_c t$$

(Source: https://ccrma.stanford.edu/software/snd/snd/fm.html)

Notice how the sum starts at "$n = 1$" and the sin term, feedback FM interestingly is the only form of FM that has its spectrum grows only toward the right/positive side instead of the both sides we have been seeing with other forms of FM. And with the spectrum being scaled by $2/nB$, the spectrum will also decay as it goes toward infinity meaning that feedback FM interestingly can be used to generate a sawtooth wave. By changing the modulation index/feedback amount B, feedback FM can be used to simulate a low pass filter sweep. Another detail is that when B is greater than 1, a burst of noise starts to form in the downward part of the waveform. As B increases, the more the output waveform resembles pure noise.

Analysis

Unintuitiveness of FM Synthesis

After fully understanding how FM synthesis generates harmonics and as well as drawing my own experiences from working with synthesizers as a musician, I came up with 2 main points on what makes FM synthesis so challenging to understand and program.

1. It is really difficult to explain how FM synthesis works fully without prior knowledge from ECE or Signals and System: While it is not uncommon for musicians to learn some ECE background knowledge like understanding how the frequency domain works for common tools like EQs and filters, understanding FM synthesis fully requires understanding additional ECE and signal processing concepts such as Hermitian symmetry, Fourier transform, Fourier series, and etc. Additionally, usually the frequency domain representation in music tools left out phase, information on the negative frequency side, and the fact Fourier transform's amplitudes are technically complex numbers, meaning most musicians, whether they have prior experience with sound synthesis and production or not, do not have natural intuition that the negative frequencies in frequency can in fact affect the positive frequencies. This concept for instance was a key idea to understand "negative frequencies reflecting back to the positive frequencies" part of FM synthesis. This is perhaps the biggest hurdle I can identify for most people including myself when it comes to learning and understanding FM synthesis.

2. Fundamental frequencies in FM synthesis often change along with the harmonics at the same time: The fundamental frequency of a harmonic spectrum is often the lowest audible frequency and the perceived pitch of that tone. Usually for most synthesis method like subtractive synthesis, additive synthesis, or even more esoteric method like phase distortion synthesis, when trying to shape or generate harmonics like changing the cutoff frequency of the filter for subtractive synthesis or adding or removing harmonics in additive synthesis, the fundamental frequency and hence the perceived pitch stays the same throughout the process. For FM synthesis, since the sidebands grow in both ways away from the carrier frequency, when changing the harmonics through parameters like the output level or the frequency ratio of the modulator operator, the fundamental frequency/perceived pitch will also change most of the time. This means that FM synthesis often changes 2 key aspects of a sound at the same time

instead of just 1 aspect at a time, which adds a level of complexity for programming a FM synthesizer.

DX7 Interface Design Analysis

As part of my completion of my capstone project, I have also decided to learn how to program my Yamaha DX7iiD directly with my knowledge of FM synthesis to better understand what interface designs work, especially with the DX7 also having the first commercially available interface to program FM synthesis.

Some of the issues with DX7 interface that I identified are

1. Only one parameter can be changed at once since there is only a single slider to adjust the value. This problem is made worse because you also need to navigate the cursor to the right parameter before the slider can change the parameter's value meaning it takes on average 2 to 3 moves before a single parameter can be accessed. However, with that said, it is also understandable why DX7 went with this interface choice. One of the main issues of FM synthesis compared to other synthesis methods is that since each operator has identical parameters, there are a lot of duplicating parameters. Additionally, DX7 also has a lot more finer controls that aren't available in the analog synthesizers that came before it. All together, if DX7 had gone with the design of representing each parameter with a single knob, the programming interface would have been so big that it wouldn't remotely fit into the size of a keyboard.
2. DX7's LCD only displays parameters as numbers. This problem became the most bothersome for me when I am adjusting the envelope shape for the operators. Since the display only shows the envelope as a series of numbers, I have to visualize this envelope myself. That is not to say other synthesizers, especially analog ones have visualization tools like a screen for the envelope but most of these synthesizers also have knobs to physically represent the values the envelope is set to, making it easier to visualize than the DX7's programming process.
3. During performance the interface is designed to change between patches rather than giving you any ability to fine tune or modify the patches half way. DX7 doesn't provide any quick way to adjust internal parameters of a patch. This means for instance if I have found the delay of patch's envelope to last just a bit too long, I will have to set the DX7 in editing mode, menu dive to find the delay time parameter, and then finally adjust that delay time parameter. DX7's interface design assumes changing or modifying sounds mean changing directly the patch you are on even though a lot of time what musicians desire is just a small change to the patch you are using.
4. Modulation Index is not displayed anywhere (even in the manual). This is an intentional design feature DX7 did to maintain the abstraction of operators. This wouldn't have been an issue if it wasn't for the fact that the curve that relates modulator carrier output to modulation index is not standardized among FM synthesizers and even the original

DX-series synthesizers. This means that you can get 2 DX series synthesizers like a DX7 and a DX21 and have the same identical parameters, they will still sound slightly different because they have slightly different modulation index.

With that said, I was also pleasantly surprised that a couple aspects of the interface were a lot more intuitive than I first thought. Some of the interface design that I came to appreciate are:

1. Despite having lots of menus to navigate through to find the right parameters, the menus and their shortcuts are actually very well organized. The same buttons that select patches on a DX7 are also used to navigate the different menus. Even though clicking through them is a chore, Yamaha engineers still added several useful shortcuts like being able to select which operator you are changing directly and being able to toggle an operator on or off by directly pressing these menu buttons. I was surprised how fast I got with programming my DX7, especially once I got used to programming with these shortcuts.
2. Even though DX7 interface prioritizing the ability to change patches quickly does sacrifice its ability to adjust parameters directly, that same ability to change patches quickly was also more useful than I first thought. In total, DX7 has 64 patches it can store natively and it uses 32 buttons to represent the selection of these with an additional button to let you toggle between patch 1 - 32 to patch 33 - 64. This means that to change a patch often just requires 1 - 2 button presses, which is a lot faster than changing a patch on most other synthesizers.

Further Works in the Future

The main question I still can't fully address which is also key in creating a good FM synthesis interface is that FM synthesis can be approached from either a complete technical method or a more exploratory approach. Personally, programming FM synthesis only started to be more intuitive once I fully understood its theory as it allowed me to somewhat predict what my outcomes were before changing a parameter. The problem is that as discussed earlier this method also does require some prior knowledge in ECE or Signals and Systems. The more exploratory approach to FM is just to let the somewhat unpredictable nature of FM to guide the sound design process. This wasn't a process as suited for me as this method requires a lot more of just testing every single possible configuration to figure out what you want, but from my exploration of trying to see how others approach FM synthesis, I have noticed that this is an approach that many musicians like or prefer.

The issue with FM synthesis is that it has a lot of parameters to control meaning there are also a lot of ways to lay out its interface, and designing an interface for it is often a question of what aspects of intuitiveness you would want to sacrifice due to physical design constraints. The main challenge is to find out what interface most people would find acceptable, and from lessons I learned from DX7's interface design, prioritizing a single preferred way to use the synthesizer generally doesn't make the synthesizer as approachable.This is certainly a project I want to continue working on in the future. With the research and work I have done for this capstone, I

am able to understand a lot more about the specifics of FM synthesis that make it hard to approach for most musicians.

Completion

To mark the completion of my capstone project, I decided to use my knowledge of FM synthesis and the DX7 to design my own 2 sound patches. This was also done such that I would have a full hands on experience with programming DX7 and importantly it has helped me identify interface design that I don't like as much and even helped me discover interface design features I ended up finding myself enjoying. As part of the submission of the capstone, I will include 2 recorded sound demos of these 2 patches that I created.

Acknowledgements

This project was supported in part by funding from the Carnegie Mellon University Frank-Ratchye Further Fund and the BXA Capstone Grant.