Introduction to Computer Music

Week 3

Version 1, 10 Sep 2018

Roger B. Dannenberg

Topics Discussed: **Sampling Theory, Perfect Sampling, Imperfect Sampling, Dither and Oversampling, Frequency Domain, Amplitude Modulation**

1 Sampling Theory Introduction

The principles which underlie almost all digital audio applications and devices, be they digital synthesis, sampling, digital recording, or CD or iPod playback, are based on the basic concepts presented in this chapter. New forms of playback, file formats, compression, and storage of data are all changing on a seemingly daily basis, but the underlying mechanisms for converting real-world sound into digital values, manipulating those data, and finally converting them back into real-world sound, has not varied much since Max Mathews developed MUSIC I in 1957 at Bell Labs.

The theoretical framework that enabled musical pioneers such as Max Mathews to develop digital audio programs stretches back over a century. For the brevity of this introduction, a mention of the groundbreaking theoretical work done at Bell Labs in the mid-20th century is worth noting. Bell Labs was concerned with transmitting larger amounts of voice data over existing telephone lines than could normally be sent with analog transmissions, due to bandwidth restrictions. Many of the developments pertaining to computer music were directly related to work on this topic.

Harry Nyquist, a Swedish-born physicist, in Certain Topics in Telegraph Transmission Theory (Trans. AIEE, vol. 47, pp. 617-644, Apr. 1928), laid out the principles for sampling analog signals at even intervals of time and at twice the rate of the highest frequency so they could be transmitted over telephone lines as "digital" signals, even though the technology to do so did not exist at the time. Part of this work is now known as the Nyquist Theorem. Nyquist worked for AT&T, then Bell Labs. Twenty years later, Claude Shannon, mathematician and early computer scientist, also working at Bells Labs and then M.I.T., developed a proof for the Nyquist theory [1]. The importance of their work to information theory, computing, networks, digital audio, digital photography and computer graphics (images are just signals in 2D!) cannot be understated. Pulse Code Modulation (PCM), a widely used method for encoding and decoding binary data, such as that used in digital audio, was also developed early on at Bell Labs, attributed to John R. Pierce, a brilliant engineer in many fields, including computer music.

In this chapter, we will learn the basic theory of *sampling*, or representing continuous functions with discrete data.

1.1 Intuitive Approach

The world is continuous. Time marches on and on, and there are plenty of things that we can measure at any instant. For example, weather forecasters might keep an ongoing record of the temperature or barometric pressure. If you are in the hospital, the nurses might be keeping a record of your temperature, your heart rate, or your brain waves.

[1]The Shannon Theorem (A mathematical theory of communication, Bell System Technical Journal, vol. 27, pp. 379-423 and 623-656, (July and October, 1948)), a pioneering work in information theory, should not be confused with the Shannon Juggling Theory of the same author, in which he worked out the mathematics of juggling numerous objects ((F+D)H=(V+D)N, where F is the time an object spends in the air, D is the time a object spends in a hand, V is the time a hand is vacant, N is the number of objects juggled, and H is the number of hands)–he was an avid juggler as well.

Any of these records gives you a function $f(t)$ where, at a given time t, $f(t)$ is the value of the particular value that interests you.

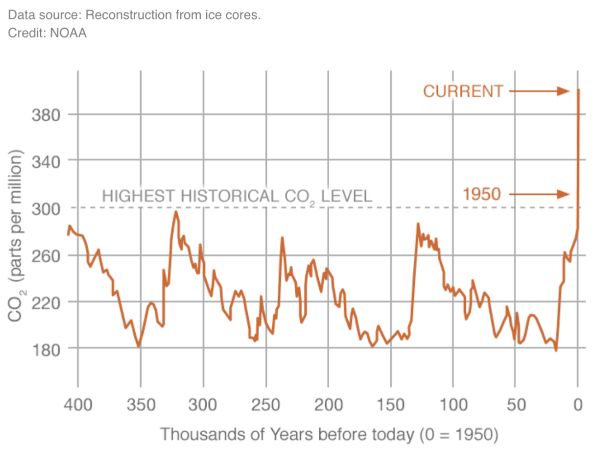

Figure 1: Carbon dioxide concentration data from climate.nasa.gov.

Figure 1 represents a continuous function of time. What we mean by "continuous" is that at any instant of time the function takes on a well-defined value, so that it makes a squiggly line graph that could be traced without the pencil ever leaving the paper. This might also be called an analog function. Of course, the time series that interest us are those that represent sound. In particular, we want to take these time series, stick them on the computer, and play with them!

Now, if you've been paying attention, you may realize that at this moment we're in a bit of a bind: sound is a continuous function. That is, at every instant in time, we could write down a number that is the value of the function at that instant——how much your eardrum has been displaced, the instantaneous air pressure, voltage from a microphone, and so on. But such a continuous function would provide an infinite list of numbers (any one of which may have an infinite expansion, like = 3.1417...), and no matter how big your computer is, you're going to have a pretty tough time fitting an infinite collection of numbers on your hard drive.

So how do we do it? How can we represent sound as a finite collection of numbers that can be stored efficiently, in a finite amount of space, on a computer, and played back and manipulated at will? In short, how do we represent sound digitally? That's the problem that we'll investigate in this chapter.

Somehow we have to come up with a finite list of numbers that does a good job of representing our continuous function. We do it by sampling the original function, at every few instants (at some predetermined rate, called the sampling rate), recording the value of the function at that moment. For example, maybe we only record the temperature every 5 minutes. The graph in Figure 1 really comes from discrete samples obtained from ice cores, so this is also an example of "sampling" to represent a continuous function using discrete values. For sound we need to go a lot faster, and we often use a special device that grabs instantaneous amplitudes at rapid, audio rates (called an analog to digital converter, or ADC).

A continuous function is also called an analog function, and, to restate the problem, we have to convert analog functions to lists of samples, in binary, which is the fundamental way that computers store information. In computers, think of a digital sound as a function of location in computer memory. That is, sounds are stored as lists of numbers, and as we read through them we are basically creating a discrete function of time of individual amplitude values.

In analog to digital conversions, continuous functions (for example, air pressures, sound waves, or voltages) are sampled and stored as numerical values. (See Figure 2.) In digital to analog conversions, numerical values are interpolated by the converter to force some continuous system (such as amplifiers, speakers, and subsequently the air and our ears) into a continuous vibration. Interpolation just means smoothly transitioning across the discrete numerical values.

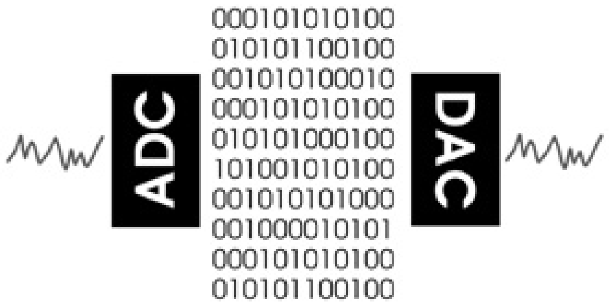

Figure 2: A pictorial description of the recording and playback of sounds through an ADC/DAC.

1.2 Analog Versus Digital

The distinction between analog and digital information is fundamental to the realm of computer music (and in fact computer anything!). In this section, we'll offer some analogies, explanations, and other ways to understand the difference more intuitively.

Imagine watching someone walking along, bouncing a ball. Now imagine that the ball leaves a trail of smoke in its path. What does the trail look like? (See Figure 3.) Probably some sort of continuous zigzag pattern, right?

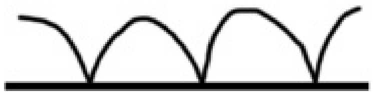

Figure 3: The path of a bouncing ball.

OK, keep watching, but now blink repeatedly. What does the trail look like now? Because we are blinking our eyes, we're only able to see the ball at discrete moments in time. It's on the way up, we blink, now it's moved up a bit more, we blink again, now it may be on the way down, and so on. We'll call these snapshots samples because we've been taking visual samples of the complete trajectory that the ball is following. (See Figure 4.) The rate at which we obtain these samples (blink our eyes) is called the sampling rate.

It's pretty clear, though, that the faster we sample, the better chance we have of getting an accurate picture of the entire continuous path along which the ball has been traveling.

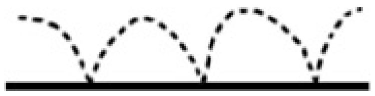

Figure 4: The same path, but sampled by blinking.

What's the difference between the two views of the bouncing ball: the blinking and the nonblinking? Each view pretty much gives the same picture of the ball's path. We can tell how fast the ball is moving and how high it's bouncing. The only real difference seems to be that in the first case the trail is continuous, and in the second it is broken, or discrete. That's the main distinction between analog and digital representations of information: analog information is continuous, while digital information is not.

1.3 Analog and Digital Waveform Representations

Now let's take a look at two time domain representations of a sound wave, one analog and one digital, in Figure 5.

The gray "staircase" waveform in Figure 5 emphasizes the point that digitally recorded waveforms are apparently missing some original information from analog sources. However, by increasing the number of samples taken each second (the sample rate), as well as increasing the accuracy of those samples (the resolution), an extremely

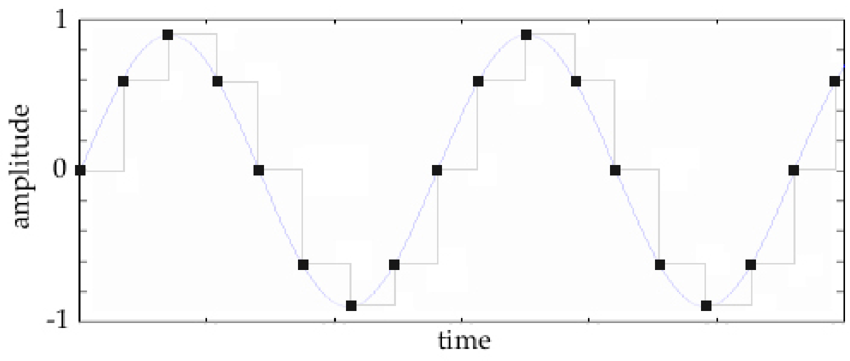

Figure 5: An analog waveform and its digital cousin: the analog waveform has smooth and continuous changes, and the digital version of the same waveform has a stairstep look. The black squares are the actual samples taken by the computer. Note that the grey lines are only for show – all that the computer knows about are the discrete points marked by the black squares. There is nothing in between those points stored in the computer.

accurate recording can be made. In fact, we can prove mathematically that discrete samples can represent certain continuous functions *without any loss of information*! And in practice, we can use this theory to record audio with imperceptible loss or distortion.

2 Sampling Theory

So now we know that we need to sample a continuous waveform to represent it digitally. We also know that the faster we sample it, the better. But this is still a little vague. How often do we need to sample a waveform in order to achieve a good representation of it?

The answer to this question is given by the Nyquist sampling theorem, which states that to well represent a signal, the sampling rate (or sampling frequency – not to be confused with the frequency content of the sound) needs to be at least twice the highest frequency contained in the sound of the signal.

2.1 Frequency Content

We'll be giving a much more precise description of the frequency domain later, but for now we can simply think of sounds as combinations of more basic sounds that are distinguished according to their "brightness." We then assign numbers to these basic sounds according to their brightness. The brighter a sound, the higher the number.

As we learned earlier, this number is called the frequency, and the basic sound is called a sinusoid, the general term for sinelike waveforms. So, high frequency means high brightness, and low frequency means low brightness (like a deep bass rumble), and in between is, well, simply in between.

All sound can ve viewed as a combination of these sinusoids, of varying amplitudes. It's sort of like making soup and putting in a bunch of basic spices: the flavor will depend on how much of each spice you include, and you can change things dramatically when you alter the proportions. The sinusoids are our basic sound spices! The complete description of how much of each of the frequencies is used is called the spectrum of the sound.

Since sounds change over time, the proportions of each of the sinusoids changes too. Just like when you spice up that soup, as you let it boil the spice proportion may be changing as things evaporate.

2.2 Highest Frequency in a Sound

Consider Figure 6. The graph at the top is our usual time domain graph, or audiogram, of the waveform created by a five-note whistled melody. Time is on the x-axis, and amplitude is on the y-axis.

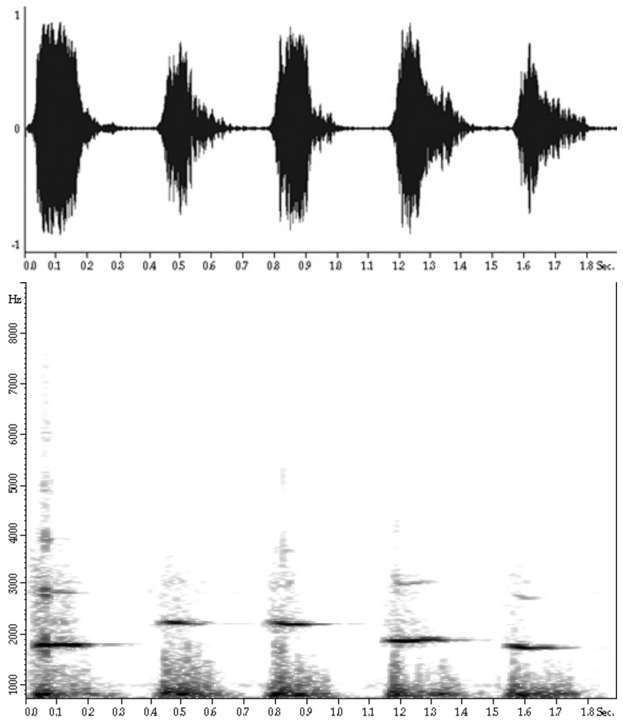

Figure 6: Two graphical representations of sound.

The bottom graph is the same melody, but this time we are looking at a time-frequency representation. The idea here is that if we think of the whistle as made up of contiguous small chunks of sound, then over each small time period the sound is composed of differing amounts of various pieces of frequency. The amount of frequency y at time t is encoded by the brightness of the pixel at the coordinate (t, y). The darker the pixel, the more of that frequency at that time. For example, if you look at time 0.4 you see a band of white, except near 2,500, showing that around that time we mainly hear a pure tone of about 2,500 Hz, while at 0.8 seconds, there are contributions all around from about 0 Hz to 3,000 Hz, but stronger ones at about 2,500 Hz and 200 Hz.

It looks like the signal only contains frequencies up to 8,000 Hz. If this were the case, we would need to sample the sound at a rate of 16,000 Hz (16 kHz) in order to accurately reproduce the sound. That is, we would need to take sound bites (bytes?!) 16,000 times a second.

In the next section, when we talk about representing sounds in the frequency domain (as a combination of various amplitude levels of frequency components, which change over time) rather than in the time domain (as a numerical list of sample values of amplitudes), we'll learn a lot more about the ramifications of the Nyquist theorem for digital sound. For example, since the human ear only responds to sounds up to about 20,000 Hz, we need to sample sounds at least 40,000 times a second, or at a rate of 40,000 Hz, to represent these sounds for human consumption. You may be wondering why we even need to represent sonic frequencies that high (when the piano, for instance, only goes up to the high 4,000 Hz range in terms of *pitches* or repetition rate.). The answer is timbral, particularly spectral. Those higher frequencies do exist and fill out the descriptive sonic information.

Just to review: we measure frequency in cycles per second (cps) or Hertz (Hz). The frequency range of human hearing is usually given as 20 Hz to 20,000 Hz, meaning that we can hear sounds in that range. Knowing that, if we decide that the highest frequency we're interested in is 20 kHz, then according to the Nyquist theorem, we need a sampling rate of at least twice that frequency, or 40 kHz.

3 The Frequency Domain

Earlier, we talked about the basic atoms of sound——the sine wave, and about the function that describes the sound generated by a vibrating tuning fork. In this section we're talking a lot about the frequency domain. If you remember, our basic units, sine waves, only had two parameters: amplitude and frequency. It turns out that these dull little sine waves are going to give us the fundamental tool for the analysis and description of sound, and especially for the digital manipulation of sound. That's the frequency domain: a place where lots of little sine waves are our best friends.

But before we go too far, it's important to fully understand what a sine wave is, and it's also wonderful to know that we can make these simple little curves ridiculously complicated, too. And it's useful to have another model for generating these functions. That model is called a phasor.

3.1 Description of a Phasor

Think of a bicycle wheel suspended at its hub. We're going to paint one of the spokes bright red, and at the end of the spoke we'll put a red arrow. We now put some axes around the wheel—the x-axis going horizontally through the hub, and the y-axis going vertically. We're interested in the height of the arrowhead relative to the x-axis as the wheel—our phasor—spins around counterclockwise. (See Figure 7.)

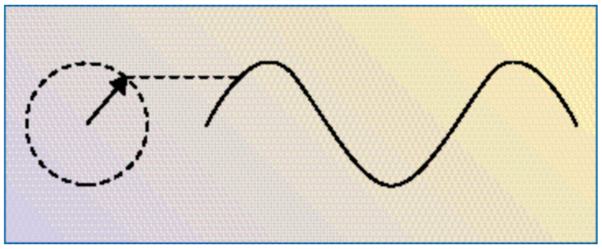

Figure 7: Sine waves and phasors. As the sine wave moves forward in time, the arrow goes around the circle at the same rate. The height of the arrow (that is, how far it is above or below the x-axis) as it spins around in a circle is described by the sine wave.

In other words, if we trace the arrow's location on the circle and measure the height of the arrow on the y-axis as our phasor goes around the circle, the resulting curve is a sine wave!

As time goes on, the phasor goes round and round. At each instant, we measure the height of the dot over the x-axis. Let's consider a small example first. Suppose the wheel is spinning at a rate of one revolution per second. This is its frequency (and remember, this means that the period is 1 second/revolution). This is the same as saying that the phasor spins at a rate of 360 degrees per second, or better yet, 2π radians per second (if we're going to be mathematicians, then we have to measure angles in terms of radians). So 2π radians per second is the angular velocity of the phasor.

This means that after 0.25 second the phasor has gone $\pi/2$ radians (90 degrees), and after 0.5 second it's gone π radians or 180 degrees, and so on. So, we can describe the amount of angle that the phasor has gone around at time t as a function, which we call $\theta(t)$.

Now, let's look at the function given by the height of the arrow as time goes on. The first thing that we need to remember is a little trigonometry.

The sine and cosine of an angle are measured using a right triangle. For our right triangle, the sine of θ, written $sin(\theta)$ is given by the equation: $sin(\theta) = a/c$ (See Figure 8.)

This means that: $a = c \times sin(\theta)$

We'll make use of this in a minute, because in this example a is the height of our triangle.

Similarly, the cosine, written $cos(\theta)$, is: $cos(\theta) = b/c$

This means that: $b = c \times cos(\theta)$

This will come in handy later, too.

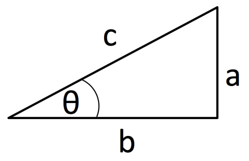

Figure 8: Triangle.

Now back to our phasor. We're interested in measuring the height at time t, which we'll denote as $h(t)$. At time t, the phasor's arrow is making an angle of $\theta(t)$ with the x-axis. Our basic phasor has a radius of 1, so we get the following relationship: $h(t) = sin(\theta(t)) = sin(2\pi t)$. We also get this nice graph of a function, which is our favorite old sine curve.

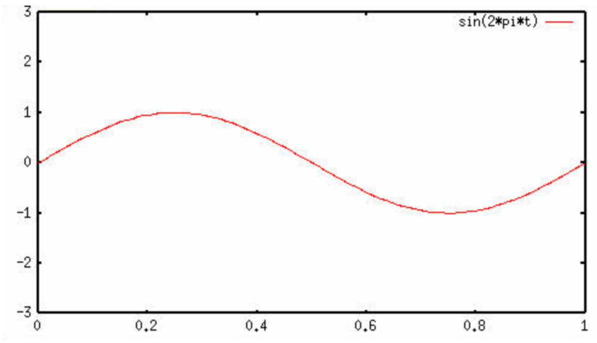

Figure 9: Basic sinusoid.

Now, how could we change this curve? Well, we could change the amplitude—this is the same as changing the length of our arrow on the phasor. We'll keep the frequency the same and make the radius of our phasor equal to 3. Then we get: $h(t) = 3 \times sin(2\pi t)$ Then we get the nice curve in Figure 10, which is another kind of sinusoid (bigger!).

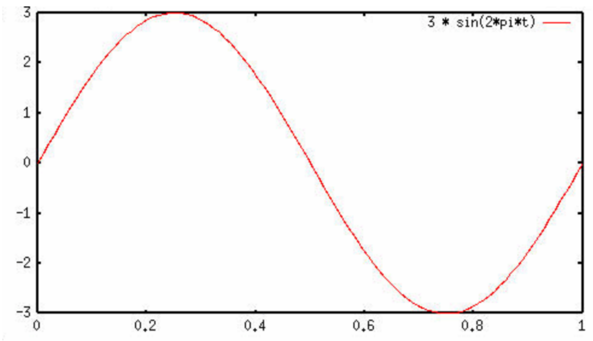

Figure 10: Bigger sine curve.

Now let's start messing with the frequency, which is the rate of revolution of the phasor. Let's ramp it up a notch and instead start spinning at a rate of five revolutions per second. Now: $\theta(t) = 5 \times (2\pi t) = 10\pi t$. This is easy to see since after 1 second we will have gone five revolutions, which is a total of 10 radians. Let's suppose that the radius of the phasor is 3. Again, at each moment we measure the height of our arrow (which we call $h(t)$), and we get: $h(t) = 3 \times sin(\theta(t)) = 3 \times sin(10\pi t)$. Now we get the sinusoid in Figure 11.

In general, if our phasor is moving at a frequency of v revolutions per second and has radius A, then plotting

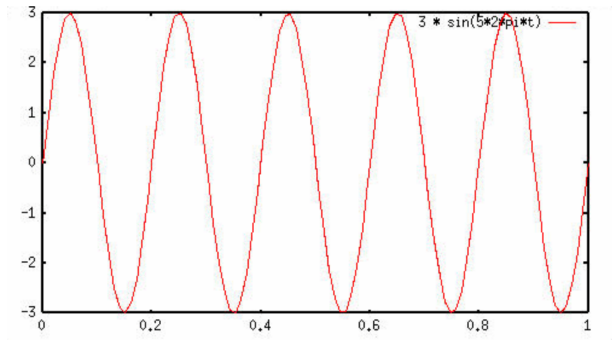

Figure 11: Bigger, faster sine curve.

the height of the phasor is the same as graphing this sinusoid: $h(t) = A \times sin(v \times 2\pi t)$. Now we're almost done, but there is one last thing we could vary: we could change the place where we start our phasor spinning. For example, we could start the phasor moving at a rate of five revolutions per second with a radius of 3, but start the phasor at an angle of $\pi/4$ radians, instead.

Now, what kind of function would this be? Well, at time t = 0 we want to be taking the measurement when the phasor is at an angle of $\pi/4$, but other than that, all is as before. So the function we are graphing is the same as the one above, but with a phase shift of $\pi/4$. The corresponding sinusoid is: $h(t) = 3 \times sin(10\pi t + \pi/4)$, which is shown in Figure 12.

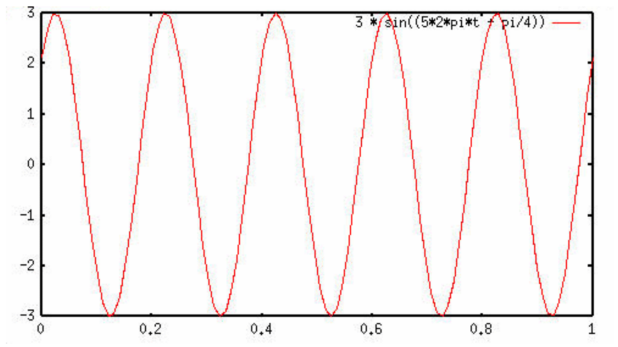

Figure 12: Changing the phase.

Our most general sinusoid of amplitude A, frequency v, and phase shift ϕ has the form: $h(t) = A \times sin(v \times 2\pi t + \phi)$, which is more commonly expressed with frequency ω in *radians per second* instead of frequency v in *cycles per second*: $h(t) = A \times sin(\omega t + \phi)$.

A particularly interesting example is what happens when we take the phase shift equal to 90 degrees, or $\pi/2$ radians. Let's make it nice and simple, with frequency equal to one revolution per second and amplitude equal to 1 as well. Then we get our basic sinusoid, but shifted ahead $\pi/2$. Does this look familiar? This is the graph of the cosine function!

You can do some checking on your own and see that this is also the graph that you would get if you plotted the displacement of the arrow from the y-axis. So now we know that a cosine is a phase-shifted sine!

3.2 Fourier Analysis

Now we know that signals can be decomposed into a (possibly infinite) sum of phasors or partials or sinusoids—however you want to look at it. But how do we determine this sum?

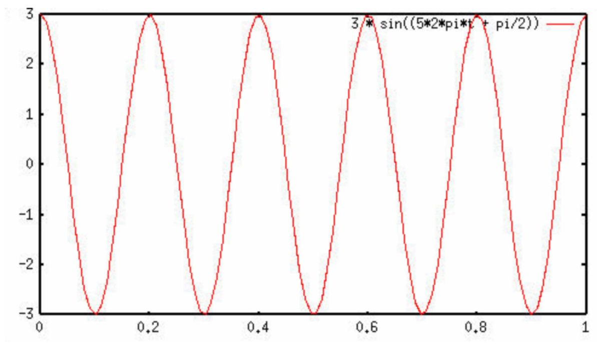

Figure 13: 90-degree phase shift (cosine).

The Fourier Transform, named for French scientist and mathematician Jean Baptiste Fourier (1768–1830), is a surprisingly simple and direct formula to write this infinite sum. Since we have to deal with phase somehow, we will begin with an expression of the Fourier Transform that keeps the sine and cosine parts separate. The signal we want to analyze is $f(t)$. The result of the analysis will be two functions. $R(\omega)$ is a function of frequency (in radians per second) that gives us the amplitude of a cosine at that frequency, and $X(\omega)$ is a function of frequency that gives us the amplitude of a sinusoid at that frequency.

$$R(\omega) = \int_{-\infty}^{\infty} f(t) cos(\omega t) dt$$

$$X(\omega) = -\int_{-\infty}^{\infty} f(t) sin(\omega t) dt$$

Why the minus sign when computing X? It relates to whether the positive direction of rotation is considered clockwise or counterclockwise. You would think the convention would be based on the simplest form (so drop the minus sign), but the Fourier Transform is closely related the the Inverse Fourier Transform (how to we construct a time domain signal from it's frequency domain representation?), and one or the other transform has to have a minus sign, so this is the convention.

Note that this is all *continuous* math. We will get back to sampling soon, but if we want to understand what sampling does to a continuous signal, we have to deal with continuous functions.

Note also, *and this is very important*, that we are "integrating out" all time. The functions R and X are functions of frequency. It does not make sense (at this point) to say "What is the Fourier Transform of the signal at time t?"—instead, we transform infinite functions of time ($f(t)$) into infinite functions of frequency, so all notions of time are absorbed into frequencies.

3.2.1 What About Phase?

We saw in the previous section how cosine is simply a sine function that is "phase shifted," and we can achieve any phase by some combination of sine and cosine at the same frequency. Thus, we get to choose: Do we want to talk about sinusoids with frequency, amplitude and phase? Or do we want to talk about cosines and sines, each with just frequency and amplitude? The two are equivalent, so we started with cosines and sines. Once we have the cosine $R(\omega)$ and sine $X(\omega)$ parts, we can derive a representation in terms of phase. For any given ω of interest, we have:

$$R(\omega) \times cos(\omega t) + X(\omega) \times sin(\omega t) = \sqrt{R(\omega)^2 + X(\omega)^2} \times sin(\omega t + arctan(X(\omega)/R(\omega)))$$

In other words, we can write the partial with frequency ω as $A(\omega) \times sin(\omega t + \theta(\omega))$ where

$$A(\omega) = \sqrt{R(\omega)^2 + X(\omega)^2}$$

and

$$\theta(\omega) = arctan(X(\omega)/R(\omega)))$$

3.2.2 Complex Representation

Rather than separating the sine and cosine parts, we can use complex numbers because, among other properties, each complex number has a real part and an imaginary part. We can think of a phasor as rotating in the complex plane, so a single complex number can represent both amplitude and phase:

$$F(\omega) = R(\omega) + j \cdot X(\omega)$$

Here, we use j to represent $\sqrt{-1}$. You probably learned to use i for imaginary numbers, but i is also the electrical engineering symbol for *current* (usually in units of Amperes), and since digital signal processing falls largely into the domain of electrical engineering, we go along with the EE's and replace i by $=j$.

Now things get interesting... Recall Euler's formula[2]:

$$e^{jx} = cos(x) + j \cdot sin(x)$$

If we rewrite the pair R, X as $R + j \cdot X$, and use Euler's formula, we can derive:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

3.2.3 Example Spectrum

The result of a Fourier Transform is a continuous function of frequency sometimes called the *frequency spectrum* or just *spectrum*. Figure 14 shows the spectrum measured from a gamelan tone. You can see there is energy in a wide range of frequencies, but also some peaks that indicate strong sinusoidal partials here and there.

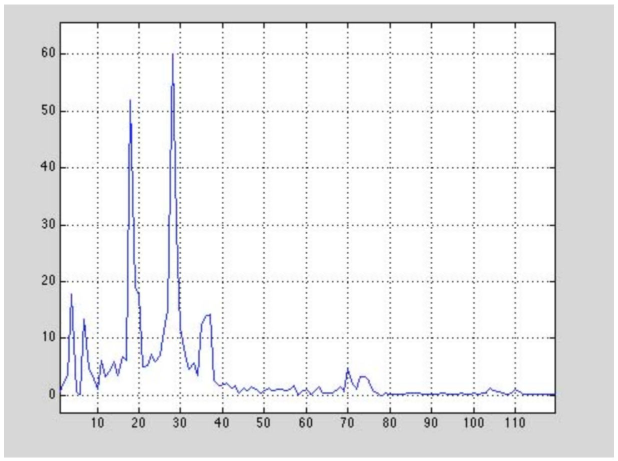

Figure 14: FFT plot of a gamelan instrument.

3.3 Fourier and the Sum of Sines

In this section, we'll try to give some intuition into the spectrum, building on the previous section.

A long time ago, Fourier proved the mathematical fact that any periodic waveform can be expressed as the sum of an infinite set of sine waves. The frequencies of these sine waves must be integer multiples of some fundamental frequency.

[2]Which one is Euler's formula? There are no less than seven formulas named for Euler, not to mention conjectures, equations, functions, identities, numbers, theorems and laws. This is the one in complex analysis. Euler invented so much mathematics that many things are named for the first person that proved Euler's original idea, just to spread the fame a little. There's a whole article in Wikipedia named "List of things named after Leonhard Euler," but I digress....

In other words, if we have a trumpet sound at middle A (440 Hz), we know by Fourier's theorem that we can express this sound as a summation of sine waves: 440 Hz, 880Hz, 1,320Hz, 1,760 Hz..., or 1, 2, 3, 4... times the fundamental, each at various amplitudes. This is rather amazing, since it says that for every periodic waveform (one, by the way, that has pitch), if we know the repetition rate, we basically know the frequencies of all of its partials. (Keep in mind however, that no trumpet player will ever play an infinitely long, steady tone, and no real-life tone is truly periodic if the periodicity comes to an end. We will not worry about this for now.)

Figure 15 shows some periodic waveforms and their associated spectra:

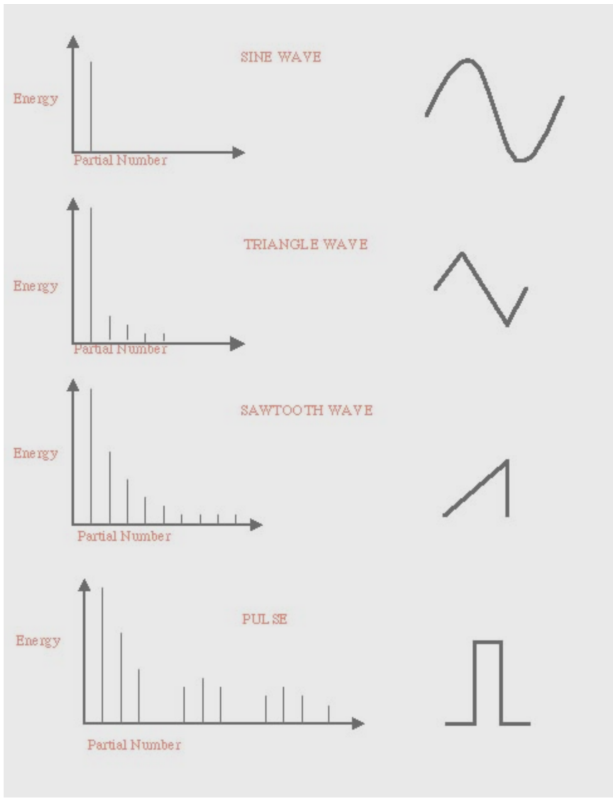

Figure 15: The spectrum of the sine wave has energy only at one frequency. The triangle wave has energy at odd-numbered harmonics (meaning odd multiples of the fundamental), with the energy of each harmonic decreasing as 1 over the square of the harmonic number ($1/N^2$). In other words, at the frequency that is N times the fundamental, we have $1/N^2$ as much energy as in the fundamental.

The partials in the sawtooth wave decrease in energy in proportion to the inverse of the harmonic number ($1/N$). Pulse (or rectangle or square) waveforms have energy over a broad area of the spectrum, but only for a brief period of time.

3.3.1 Real, Imaginary, Amplitude, Phase

In these drawing, we are carelessly forgetting about phase. Generally, we cannot hear phase differences, but we are very adept at hearing amplitudes at different frequencies. Therefore, we usually plot just the amplitude of the signal when we plot the spectrum, and we ignore the phase. Recall that in terms of real and imaginary parts, the amplitude is just the square root of the sum of squares:

$$A(\omega) = \sqrt{R(\omega)^2 + X(\omega)^2}$$

4 Sampling and the Frequency Domain

Now that we understand something about the Fourier Transform, we return to our problem of understanding sampling. What does sampling do to a signal? In the continuous math world, we can model sampling as a simple multiplication of two time domain signals, as shown in Figure 16.

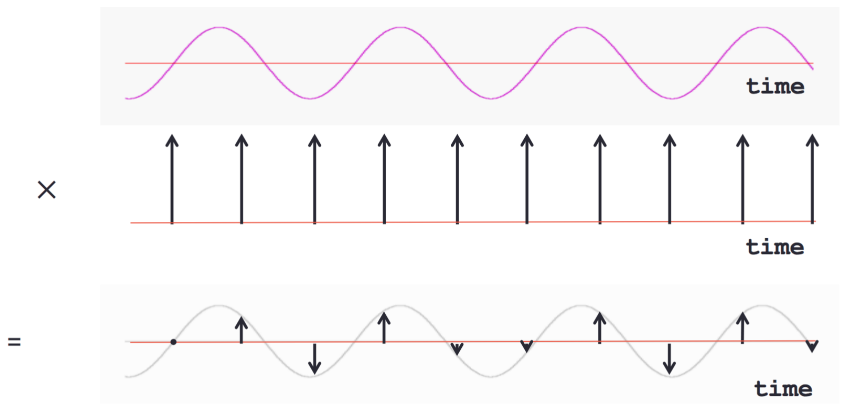

Figure 16: Sampling is effectively multiplication in the time domain: We multiply by 1 (or more properly an instantaneous but infinite impulse whose integral is 1) whereever we capture samples, and we remove the rest of the signal by multiplying by zero.

This gets quite interesting in the frequency domain. Multiplying by a series of impulses in the time domain (Figure 16) is equivalent to copying and shifting the spectrum in the frequency domain (Figure 17). The copies are shifted by the *sampling rate*, which of course is a frequency.

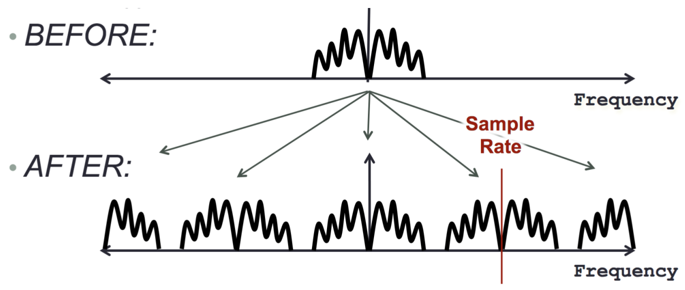

Figure 17: Sampling in the time domain gives rise to shifted spectral copies in the frequency domain. While shown as separate copies here, the copies are in fact summed into a single spectrum.

4.1 Negative Frequencies

Looking at Figure 17, you should be wondering why the spectrum is shown with negative as well as positive frequencies. You can consider negative frequencies to be just the result of the formulas for the Fourier Transform. While "negative 440 Hertz" may sound like nonsense, in mathematical terms, it is just a question of writing $sin(\omega t)$ or $sin(-\omega t)$, but $sin(-\omega t) = sin(\omega t + \pi)$, so we are really talking about certain phase shifts. Since we mostly ignore phase anyway, we often plot only the positive frequencies. In the case of sampling theory, we show both negative and positive frequencies of the spectrum.

4.2 Aliasing

Notice in Figure 17 that after sampling, the copies of the spectrum come close to overlapping. What would happen if the spectrum contained higher frequencies. Figure 18 illustrates a broader spectrum.

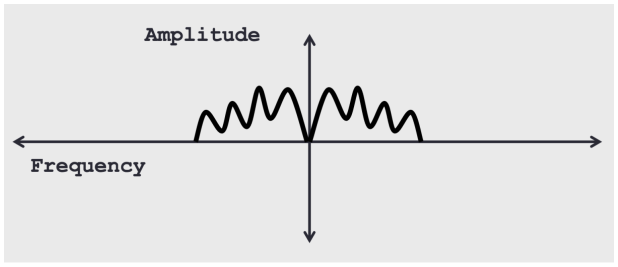

Figure 18: A broader spectrum (contains higher frequencies) before sampling.

Figure 19 illustrates the signal after sampling. What a mess! The copies of the spectrum now overlap. The Nyquist frequency (one-half the sampling rate, and one half the amount by which copies are shifted) is shown by a vertical dotted line. Notice that there is a sort of mirror symmetry around that line: the original spectrum extending above the Nyquist frequency is "folded" to become new frequencies below the Nyquist frequency.

This is really bad because new frequencies are introduced and *added* to the original spectrum. The frequencies are in fact aliases of the frequencies above the Nyquist frequency. To avoid aliasing, it is necessary to sample at higher than twice the frequency of the original signal.

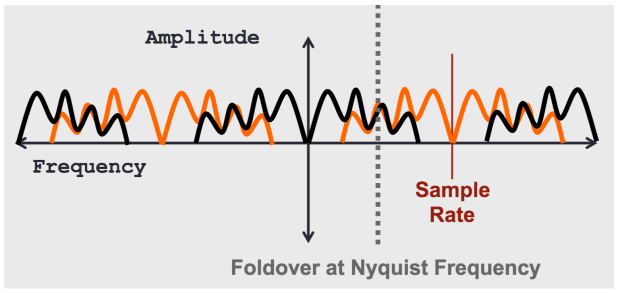

Figure 19: A broader spectrum (contains higher frequencies) before sampling.

In the time domain, aliasing is a little more intuitive, at least if we consider the special case of a sinusoid. Figure 20 shows a sinusoid (blue) at a frequency more than half the sample rate. This frequency is "folded" to become a lower frequency (red) below half the sample rate. Note that both sinusoids pass perfectly through the available samples, which is why they are called aliases. They have the same "signature" in terms of samples.

To summarize, *the frequency range (bandwidth) of a sampled signal is determined by the sample rate.* We can only capture and represent frequencies below half the sample rate, which is also called the *Nyquist rate* or *Nyquist frequency*.

5 Sampling without Aliasing

In practice, we seldom have the luxury of controlling the bandwidth of our analog signals. How do you tell the triangle player to make those tinkly sounds, but don't go above 22 kHz? The real world is full of high frequencies we cannot hear, but which can become audible as aliases if we are not careful. The solution and standard practice is to *filter* out the high frequencies *before* sampling, as shown in Figure 21.

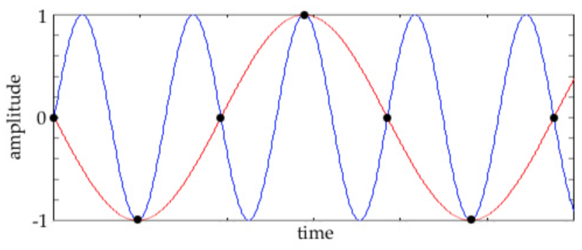

Figure 20: Aliasing, foldover. This is a phenomenon that happens when we try to sample a frequency that is more than half the sampling rate, or the Nyquist frequency. As the frequency we want to sample gets higher than half the sampling rate, we start "undersampling" and get unwanted, lower-frequency artifacts (that is, low frequencies created by the sampling process itself).

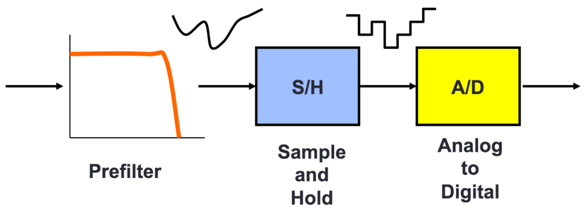

Figure 21: Aliasing is avoided by removing frequencies above the Nyquist frequency using a prefilter (sometimes called an *anti-aliasing filter*.

5.1 Converting Back to Analog

A similar process is used to convert digital signals back to analog. If we were to simply output a train of impulses based on the digital samples, the spectrum would look like that in Figure 17, with lots of very high frequencies (in fact, these are aliases of all the sinusoids in the original spectrum). We remove these unwanted parts of the signal with another filter as shown in Figure 22.

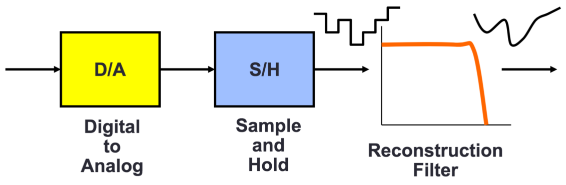

Figure 22: To convert a digital signal to analog, we need to *remove* the artifacts of sampling. This is accomplished by sending the output of the digital-to-analog converter first into a sample-and-hold circuit that holds the signal steady while the converter is transitioning to the next sample value, then through a *reconstruction filter*, which removes high frequencies above the Nyquist frequency.

6 Imperfect Sampling

So far, we have described sampling as if samples were real numbers obtained without error. What happens in the imperfect real world, and what happens to rounding errors, since real samples are actually integers with finite precision.

6.1 How to Describe Noise

To describe “error” in audio, we need a way to measure it. Since our hearing basically responds to audio in a logarithmic manner, the important thing is not the absolute error but the ratio between the signal and the error. We call errors in audio “noise,” and we measure the *signal to noise ratio*.

Ratio is so important in audio, and the range of signals and noise is so large, that we use a special unit, the dB or decibel to represent ratios, or more precisely logarithms of ratios. a Bel represents a factor of 10 in power, which is 10 decibels. Since this is a logarithmic scale, 20 decibels is two factors of 10 or a factor of 100 in power. But power increases with the square of amplitude, so 20 dB results in only a factor of 10 in amplitude. A handy constant to remember is a factor of 2 in amplitude is about 6 dB, a factor of 4 is about 12 dB. What can you hear? A change of 1 dB is barely noticeable. A change of 10 dB is easily noticeable, but your range of hearing is about 10 of these steps (around 100 dB).

6.2 Quantization Noise

What is the effet of rounding? Imagine that there is significant signal change between each pair of samples and the true value of the signal we are capturing has no preference for any value over any other. In that case, we can expect the signal to fall randomly between two integer sample values. If we round to the nearest value, the rounding error will be a uniformly distributed random number between -0.5 and 0.5 (considering samples to be integers).

Figure 23 illustrates a signal being captured and the resulting rounding error. Note that the samples must fall on the grid of discrete time points and discrete integer sample values, but the actual signal is continuous and real-valued.

Rather than thinking of the signal as being recorded with error, consider this: If we could subtract the quantization error from the original signal, then the original signal would pass through integer sample values and we could capture the signal perfectly! Thus, the effect of quantization error is to add random values in the range -0.5 to +0.5 to the original signal before sampling.

It turns out that uniform random samples are white noise, so calling error “noise” is quite appropriate. How loud is the noise? The resulting signal-to-noise ratio, given M-bit samples, is $6.02M + 1.76$ dB. This is very close to saying “the signal-to-noise ratio is 6dB per bit.” You only get this signal-to-noise ratio if the signal uses the full range of the integer samples. In real life, you always record with some “headroom,” so you probably lose 6 to 12 dB of the best possible signal. There is also a limit to how many bits you can use—remember that the bits are coming from analog hardware, so how quiet can make circuits, and how accurately can you measure voltages? In practice, 16-bit converters offering a theoretical 98 dB signal-to-noise ratio are readily available and widely used. 20- and 24-bit converters are used in high-end equipment, but these are not accurate to the least-significant bit, so do not assume the signal-to-noise ratio can be estimated merely by counting bits.

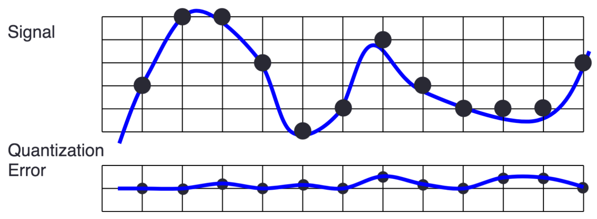

Figure 23: When a signal is sampled, the real-valued signal must be rounded at each sample to the nearest integer sample value. The resulting error is called *quantization error*.

7 Sampling Summary

One amazing result of sampling theory is that if the signal being sampled has a finite bandwidth, then we can sample the signal at discrete time points and *completely recover* the original continuous signal. It seems too good to be true that all those infinitely many values between samples are not really necessary!

Aside from this startling theoretical finding, we learned that the sample rate should be at least double that of the highest frequency of the signal. Also, error is introduced by rounding samples to integers, an unavoidable step in the real world, but the signal-to-noise ratio is about 6 dB per bit, so with 16 bits or more, we can make noise virtually imperceptible.

8 Special Topics in Sampling

8.1 Dither

When we described quantization noise, we assumed that rounding error was random. Now consider the case where the signal is a pure sine tone with amplitude from -1.4 to +1.4. It should be clear that after rounding, this signal will progress through sample values of -1, 0, 1, 0, -1, 0, 1, 0, ... (possibly repeating each of these values, depending on frequency). This is anything but random, and this non-random rounding will inject higher, possibly audible frequencies into the recorded signal rather than noise. To avoid this unpleasant phenomenon, it is common to actually add noise *on purpose* before sampling. The added noise "randomizes" the rounding so that more audible artifacts are avoided.

This process of adding noise is called *dithering*. Dithering is also used in computer graphics, photography and printing to avoid "posterized" effects when rendering smooth color gradients. Another, more sophisticated way to use dithering, also called *noise shaping*, is to design the spectrum of the added dither noise to optimize the perceived signal-to-noise ratio. Since we are more sensitive to some frequencies than others, when we add noise to a signal it is a good idea to include more hard-to-hear frequencies in the noise than easy-to-hear noise. With this trick, we can make the apparent quality of 16-bit audio *better* than 16-bits in some frequency regions where we are most sensitive, by sacrificing the signal-to-noise ratio at frequencies (especially high ones) where we do not hear so well.

8.2 Oversampling

Another great idea in sampling is oversampling. A problem with high-quality digital audio conversion is that we need anti-aliasing filters and reconstruction filters that are (1) analog and (2) have very difficult to achieve requirements, namely that the filter passes everything without alteration right up to the Nyquist frequency, then suddenly cuts out everything above that frequency. Filters, typically have a smooth transition from passing signal to cutting signals, so anti-aliasing filters and reconstruction filters used to be marvels of low-noise audio engineering and fabrication.

With oversampling, the idea is to sample at a very high sample rate to avoid stringent requirements on the analog anti-aliasing filter. For example, suppose your goal is to capture frequencies up to 20 kHz. Using a sample rate of 44.1 kHz means your anti-aliasing filter must pass 20 kHz and reject everything above 22.05 kHz (half of 44.1 kHz). This requires an elaborate and expensive filter. Instead, suppose you (over)sample at 176.4 kHz so you can capture up to 88.2 kHz. Now, your analog filter can pass everything up to 20 kHz and does not need to reject everything until 88.2 kHz, so there are about 2 octaves to make that transition, a much easier filter to build in practice.

But, now you have a 176.4 kHz sample rate, and some frequencies between 20 and 88.2 kHz leaked through your cheap filter. No problem! We can implement another filter in the digital domain to get rid of everything above 20 kHz and then simply drop 3 out of 4 samples to get down to the desired 44.1 kHz sample rate.

Similarly, we can digitally up-sample from 44.1 kHz to 176.4 kHz just before digital-to-analog conversion so that the high output frequencies start above 88.2 kHz, making it easy to build a high-quality, low-noise reconstruction (smoothing) filter.

Oversampling was invented by Paul Lanksy, a computer music composer at Princeton, and first published as a letter to Computer Music Journal. Paul did not patent his invention or anticipate that it would be used in billions of digital audio devices.

9 Amplitude Modulation

Time-domain representations show us a lot about the amplitude of a signal at different points in time. Amplitude is a word that means, more or less, “how much of something,” and in this case it might represent pressure, voltage, some number that measures those things, or even the in-out deformation of the eardrum.

For example, the time-domain picture of the waveform in Figure 24 starts with the attack of the note, continues on to the steady-state portion (sustain) of the note, and ends with the cutoff and decay (release). We sometimes call the attack and decay *transients* because they only happen once and they don’t stay around! We also use the word transient, perhaps more typically, to describe timbral fluctuations during the sound that are irregular or singular and to distinguish between those kinds of sounds and the steady state.

From the typical sound event shown in Figure 24, we can tell something about how the sound’s amplitude develops over time (what we call its amplitude envelope).

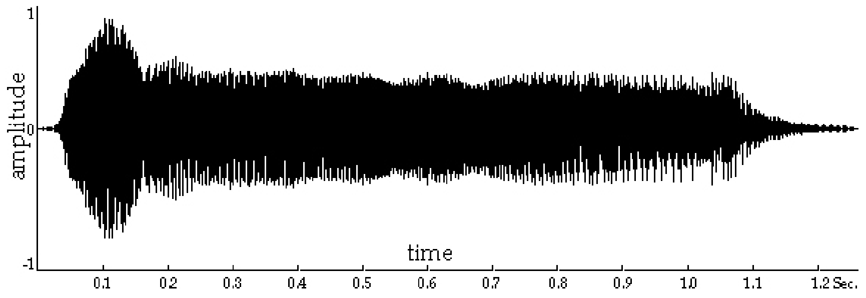

Figure 24: A time-domain waveform. It’s easy to see the attack, steady-state, and decay portions of the "note" or sound event, because these are all pretty much variations of amplitude, which time-domain representations show us quite well. The amplitude envelope is a kind of average of this picture.

9.1 Envelopes in Sound Synthesis

In sound synthesis, we often start with a steady tone such as a periodic waveform and multiply by an envelope function to create an artificial amplitude envelope. In Nyquist (the programming language), we write this using the multiply (`*`) operator in expressions that typically look like:

steady-signal-expression * *envelope-expression*

In Nyquist, you can use the `pwl` or `env` functions, for example, to make envelopes, as we have seen before.

Another type of amplitude modulation is amplitude vibrato, where we want a wavering amplitude. One way to create this effect is to vary the amplitude using a low-frequency oscillator (LFO). However, be careful not to simply multiply by the LfO. Figure 25) shows what happens to a signal multiplied by `lfo(6)`.

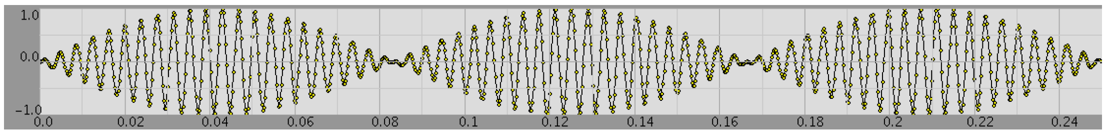

Figure 25: The output from `osc(c4) * lfo(6)`. The signal has extreme pulses at twice the LFO rate because `osc(6)` crosses through zero twice for each cycle. Only 1/4 second is plotted.

To get a more typical vibrato, offset the`lfo` signal so that it does not go through zero. Figure 26 shows an example. You can play the expressions from Figures 25 and 26 and hear the difference.

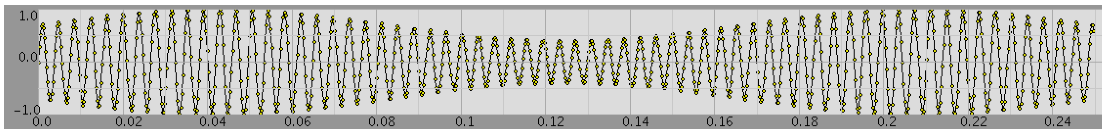
Figure 26: The output from `osc(c4) * (0.7 + 0.3 * lfo(6))`. The signal vibrates at 6 Hz and does not diminish to zero because the vibrato expression contains an offset of 0.7.

9.2 Amplitude Modulation by Audible Frequencies

If the modulation rate shown in Figure 25 is increased from 6 Hz into the audible range, some very interesting things happen. In the frequency, the original spectrum is shifted up and down by the modulation frequency. So a rapidly modulated sinusoid becomes two sinusoids, and there is no energy at the original frequency—it is all shifted. Figure 27 shows the modulation of an 880 Hz signal by a 220 Hz modulation.

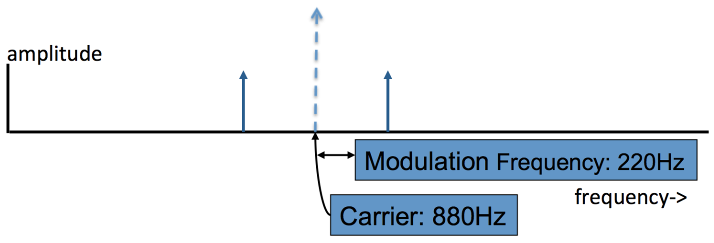

Figure 27: The result of modulating (multiplying) one sinusoid at 880 Hz by another at 220 Hz.

In more general terms, an entire complex signal can be transformed by amplitude modulation. Figure 28 shows the effect of modulating a complex signal indicated by the trapezoidal-shaped spectrum. The original spectrum (dotted lines) is shifted up and down. This effect is also known as *ring modulation*. It has a characteristic sound of making tones sound metalic because their shifted harmonic spectrum is no longer harmonic.

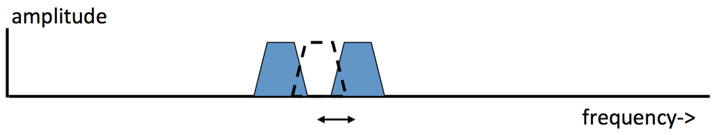

Figure 28: The result of modulating (multiplying) a complex signal (illustrated by the trapezoidal spectrum in dotted lines) by a sinusoidal modulator. The original spectrum is shifted up and down by the frequency of the modulator.

10 Acknowledgments

Thanks to Shuqi Dai and Sai Samarth for editing assistance.

Portions of this work are taken almost verbatim from *Music and Computers, A Theoretical and Historical Approach* (`http://sites.music.columbia.edu/cmc/MusicAndComputers/`) by Phil Burk, Larry Polansky, Douglas Repetto, Mary Robert, and Dan Rockmore. Other portions are taken almost verbatim from *Introduction to Computer Music: Volume One* (`http://www.indiana.edu/~eemusic/etext/toc.shtml`) by Jeffrey Hass. I would like to thank these authors for generously sharing their work and knowledge.