Practical digital audio signal processing

A gentle introduction to the FFT

Posted on August 31, 2002 by Nigel Redmon

Some terms: The Fast Fourier Transform is an algorithm optimization of the DFT—Discrete Fourier Transform. The "discrete" part just means that it's an adaptation of the Fourier Transform, a continuous process for the analog world, to make it suitable for the sampled digital world. Most of the discussion here addresses the Fourier Transform and its adaptation to the DFT. When it's time for you to implement the transform in a program, you'll use the FFT for efficiency. The results of the FFT are the same as with the DFT; the only difference is that the algorithm is optimized to remove redundant calculations. In general, the FFT can make these optimizations when the number of samples to be transformed is an exact power of two, for which it can eliminate many unnecessary operations.

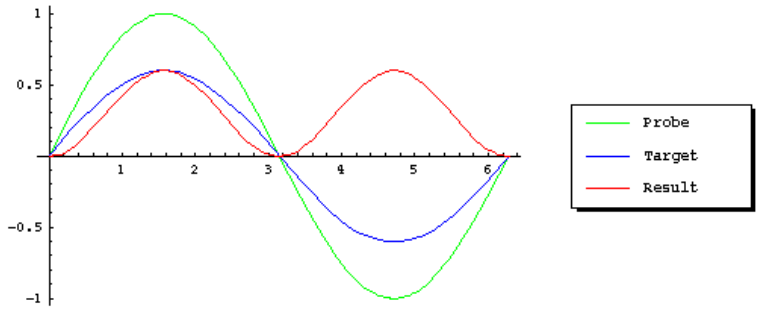
Background

From Fourier we know that periodic waveforms can be modeled as the sum of harmonically-related sine waves. The Fourier Transform aims to decompose a cycle of an arbitrary waveform into its sine components; the Inverse Fourier Transform goes the other way—it converts a series of sine components into the resulting waveform. These are often referred to as the "forward" (time domain to frequency domain) and "inverse" (frequency domain to time domain) transforms. For most people, the forward transform is the baffling part—it's easy enough to comprehend the idea of the inverse transform (just generate the sine waves and add them). So, we'll discuss the forward transform; however, it's interesting to note that the inverse transform is identical to the forward transform (except for scaling, depending on the implementation). You can essentially run the transform twice to convert from one form to the other and back!

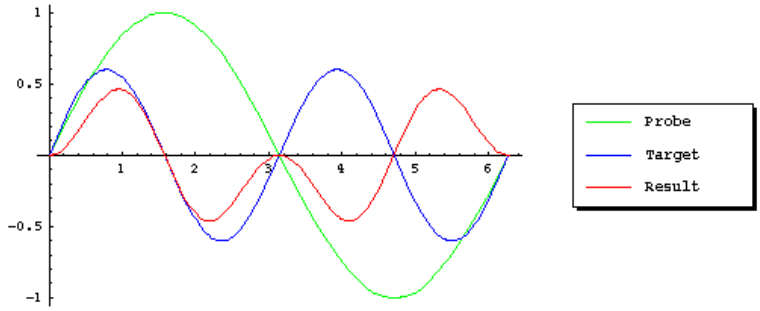
Probing for a match

Let's start with one cycle of a complex waveform. How do we find its component sine waves? (And how do we describe it in simple terms without mentioning terms like "orthogonality"? oops, we mentioned it.) We start with an interesting property of sine waves. If you multiply two sine waves together, the resulting wave's average (mean) value is proportional to the sines' amplitudes if the sines' frequencies are identical, but zero for all other frequencies.

Take a look: To multiply two waves, simply multiply their values sample by sample to build the result. We'll call the waveform we want to test the "target" and the sine wave we use to test it with the "probe". Our probe is a sine wave, traveling between -1.0 and 1.0. Here's what happens when our target and probe match:



See that the result wave's peak is the same as that of the target we are testing, and its average value is half that. Here's what happens when they don't match:



In the second example, the average of the result is zero, indicating no match.

The best part is that the target need not be a sine wave. If the probe matches a sine component in the target, the result's average will be non-zero, and half the component's amplitude.

In phase

The reason this works is that multiplying a sine wave by another sine wave is balanced modulation, which yields the sum and difference frequency sine waves. Any sine wave averaged over an integral number of cycles is zero. Since the Fourier transform looks for components that are whole number multiples of the waveform section it is analyzing, and that section is also presumed to be a single cycle, the sum and difference results are always integral to the period. The only case where the results of the modulation don't average to zero is when the two sine waves are the same frequency. In that case the difference is 0 Hz, or DC (though DC stands for Direct Current, the term is often used to describe steady-state offsets in any kind of waveform). Further, when the two waves are identical in phase, the DC value is a direct product of the multiplied sine waves. If the phases differ, the DC value is proportional to the cosine of the phase difference. That is, the value drops following the cosine curve, and is zero at pi/2 radians, where the cosine is zero.

So this sine measurement doesn't work well if the probe phase is not the same as the target phase. At first it might seem that we need to probe at many phases and take the best match; this would result in the ESFT—the Extremely Slow Fourier Transform. However, if we take a second measurement, this time with a cosine wave as a probe, we get a similar result except that the cosine measurement results are exactly in phase where the sine measurement is at its worst. And when the target phase lies between the sine and cosine phase, both measurements get a partial match. Using the identity

$$\sin(\theta)^2 + \cos(\theta)^2 = 1$$

for any theta, we can calculate the exact phase and amplitude of the target component from the sine and cosine probes. This is it! Instead of probing the target with all possible phases, we need only probe with two. This is the basis for the DFT.

Completing the series

Besides probing with our single cycle sine (and cosine), the presumed fundamental of the target wave, we continue with the harmonic series (2x, 3x, 4x...) through half the sample rate. At that point, there are only two sample points per probe cycle, the Nyquist limit. We also probe with 0x, which is just the average of the target and gives us the DC offset.

We can deduce that having more points in the "record" (the group of samples making up our target wave cycle) allows us to start with a lower frequency fundamental and fit more harmonic probes into the transform. Doubling the number of target samples (higher time resolution) doubles the number of harmonic probes (higher frequency resolution).

Getting complex

By tradition, the sine and cosine probe results are represented by a single complex number, where the cosine component is the real part and the sine component the imaginary part. There are two good reasons to do it this way: The relationship of cosine and sine follows the same mathematical rules as do complex numbers (for instance, you add two complex numbers by summing their real and complex parts separately, as you would with sine and cosine components), and it allows us to write simpler equations. So, we refer to the resulting average of the cosine probe as the real part (Re), and the sine component as the imaginary part (Im), where a complex number is represented as Re + i*Im.

To find the magnitude (which we have called "amplitude" until now—magnitude is the same as amplitude when we are only interested in a positive value—the absolute value):

$$\text{magnitude} := \sqrt{Re^2 + Im^2}$$

In the way we've presented the math here, this is the magnitude of the average, so again we'd have to multiply that value by two to get the peak amplitude of the component we're testing for.

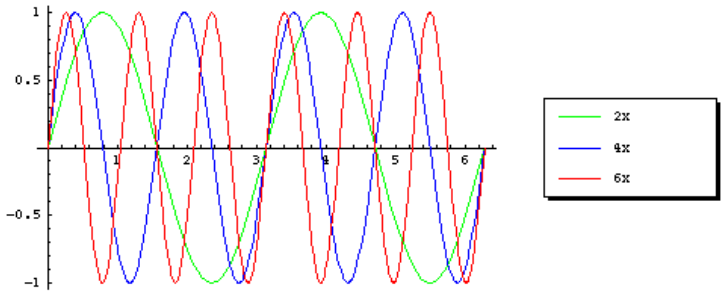
$$\text{phase} := \text{atan2}(Im, Re)$$

Many computer languages and math packages support the atan2 function. Basically, this gives the arc tangent of Im/Re, while handling the special cases of the four quadrants and divide by zero for you. This give you the phase shift of each harmonic in radians. Since the real part corresponds to cosine, you can see that a harmonic with an imaginary part of zero results in a phase of zero—corresponding to a cosine.

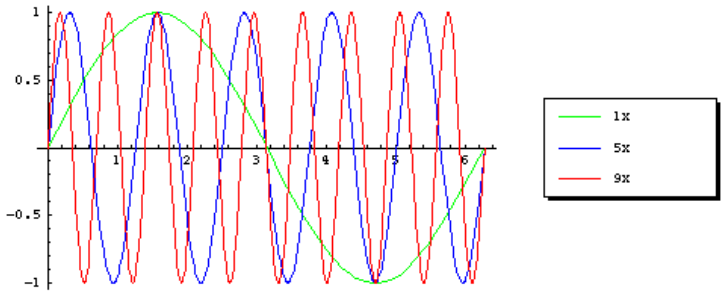
Making it "F"

Viewing the DFT in this way, it's easy to see where the algorithm can be optimized. First, note that all of the sine probes are zero at the start and in the middle of the record—no need to perform operations for those. Further, all the even-numbered sine probes cross zero at one-fourth increments through the record, every fourth probe at one-eighth, and so on. Note the powers of two in this pattern. The FFT works by requiring a power of two length for the transform, and splitting the the process into cascading groups of two (that's why it's sometimes called a radix-2 FFT). Similarly, there are patterns for when the sine and cosine are at 1.0, and multiplication is not needed. By exploiting these redundancies, the savings of the FFT over the DFT are huge. While the DFT needs N^2 basic operations, the FFT needs only NLog2(N). For a 1024 point FFT, that's 10,240 operations, compared to 1,048,576 for the DFT.

Let's take a look at the kinds of symmetry exploited by the FFT. Here's an example showing even harmonics crossing at zero for integer multiples of pi/2 on the horizontal axis:



Here we see that every fourth harmonic meets at 0, 1, 0, and -1, at integer multiples of pi/2:



Caveats and Extensions

The Fourier transform works correctly only within the rules laid out—transforming a single cycle of the target periodic waveform. In practical use, we often sample an arbitrary waveform, which may or may not be periodic. Even if the sampled waveform is exactly periodic, we might not know what that period is, and if we did it may not exactly fit our transform length (we may be using a power-of-two length for the FFT).

We can still get results with the transform, but there is some "spectral leakage." There are ways to reduce such errors, such as windowing to reduce the discontinuities at the ends of the group of sample points (where we snipped the chunk to examine from the sampled data). And for arbitrarily long signals (analyzing a constant stream of incoming sound, for instance), we can perform FFTs repeatedly—much in the way a movie is made up of a constant stream of still pictures—and overlap them to smooth out errors.

There is a wealth of information on the web. Search for terms used here, such as Fourier, FFT, DFT, magnitude, phase... The purpose here is to present the transform in an intuitive way. With an understanding that there is no black magic involved, perhaps the interested reader is encouraged to dig deeper without fear when it's presented in a more rigorous and mathematical manner. Or maybe having a basic idea of how it works is good enough to feel more comfortable with using the FFT. You can find efficient implementations of the FFT for many processors, and links to additional information, at http://www.fftw.org. For another source on the transform and basic C code, try Numerical Recipes in C.

This entry was posted in Digital Audio, FFT. Bookmark the permalink.

11 Responses to *A gentle introduction to the FFT*

Gary Frost *says:*
June 19, 2011 at 9:10 am

Thanks for this. I have been scratching my head to work out how FFT works and this provided a very good background.

Reply

Stephen *says:*
January 9, 2012 at 7:12 pm

Thanks for posting this. As a VLSI student trying to understand FFT this was very helpful in giving me a basic understanding.

Reply

Vadim *says:*
January 24, 2012 at 1:55 pm

The most clear explanation I have ever read! Must read! Thanks for the article!

Reply

Robert Macharia *says:*
July 16, 2012 at 12:14 am

wow!!! the best eye-opener i have seen in ages!!! thanks a bunch!!!

Reply

Hans Linkels *says:*
March 12, 2013 at 3:43 pm

Thank you SO much for this web page. I seen FFT being mentioned for 30 years, even used it, but never understood what the principle behind was. Now in 3 paragraphs I understood completely!

One note: I think the phase should really be arctan(Im/Re).

When Im=0 you should have a phase of 0, or the real signal coinciding with a cosine. The angle of the complex voltage = 0, so the complex voltage is fully real.

When Re=0, arctan(Im/Re) becomes infinite and the phase angle is 90 degrees, or a signal fully coinciding with a sine.

jlinkels

Reply

Nigel Redmon *says:*
March 13, 2013 at 5:20 pm

Thank you, Hans, for catching this. I can't believe it went so long with no one noticing! I'll fix the artwork…I'll change it to atan2 while I'm at it, to be more specific about quadrant…

Nigel

Reply

davorin *says:*
April 30, 2013 at 6:45 am

This explanation really worked for me. Another interesting angle is this guide that explains each sample in time as a "spike" and you can calculate how much each harmonic contributes to that spike (I hope I got that right). Still, the idea of probing with sin and cos is easier to wrap my head around.

Reply

Saket Jalan *says:*
April 22, 2014 at 5:05 am

I had been introduced to the Fourier Transform in College (about 20 years back) and have used the FFT in some way or the other since than as part of my job.

But, the intutive feel that this article gave me for Fourier Transform had somehow always escaped me. Reading this article, it hit me write in the face and I was amazed at the simplicity of it.

Many thanks.

Reply

Pete *says:*
November 14, 2017 at 5:09 pm

Hi Nigel

Best FFT description I've ever read. And I've read a few! Starting off by explaining the 'phenomenon' of sin wave multiplication giving a non-zero average when the target and probe waves have the same frequency was inspired, as it seems to be the basis of it all!

Thanks again
Pete

Reply

Bert Kraaijpoel *says:*
January 3, 2022 at 10:49 am

Thank you for offering this easy understanding of DFT and FFT.
I will share it with my students!

Bert

Reply

Nigel Redmon *says:*
January 3, 2022 at 11:17 am

Thank you, I hope it's helpful to your students! On my long to-do list is to make a video on the topic…

Reply

EarLevel Engineering