SpectrumAnalysis

Charles P. Peterson

COLLABORATORS

	TITLE : SpectrumAnalysis	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Charles P. Peterson	February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SpectrumAnalysis

1.1 Spectrum Analysis Applications

This document describes a few spectrum analysis applications, and also shows pictures illustrating the power of GFFT. For a complete discussion of the _features_ of GFFT, refer to the on-line help available while running GFFT, or read the on-line help text file, GFFT.HELP. If you are having trouble displaying the pictures click here .

If you'd like to read this entire document, you can browse all the way through it, but be sure to look at all of the pictures (which are accessed through some of the buttons in the text).

Contents:

1.2 Introduction to Spectrum Analysis

Spectrum analysis involves the conversion of a sound from the " ↩
time domain"
in which it occurs over a period of time, into the "frequency domain," which is an alternative way of thinking about it (or, maybe a parallel universe). In the time domain, a sound (or any periodic phenomenon) is a succession of amplitudes occuring over time, while in the frequency domain a sound is the summation of a set of frequencies, each having a particular amplitude which may change over time. Our ear does a kind of spectrum analysis in converting the frequencies we hear into pitch and timbre.

There are two basic kinds of applications for spectrum analysis. In the first kind,

waveform analysis

, you attempt to understand a particular periodic pattern (such as a sound, vibration, or electromagnetic signal) more completely. This may tell you more about the physical or electronic structure which mades it, the physiology of the animal which produced it, or how you might better be able to produce a synthetic version.

In the second kind,

frequency response analysis

, you attempt to evaulate, understand, and possibly improve the frequency response of a sound reproduction system or component. Ideally, such a reproducer would have uniform or "flat" frequency response. In order to measure the frequency

response, you must provide a input signal of some kind, and then compare the spectrum of the input signal with that of the output. This document focuses primarily on frequency response analysis, which, unfortunately, is a little harder to do.

1.3 Waveform Analysis

It is interesting and frequently useful to understand the harmonic structure of some sound. For example, if you knew the harmonics in a particular acoustic instrument, you could begin to synthesize its sound using a collection of electronic oscillators. (Unfortunately, there's a lot more to it than that. And please don't ask me to modify GFFT to synthesize instruments, unless you have a lot of money to spend on it.)

Real acoustic sounds frequently have a large number of 'harmonics,' which are multiples of the frequency of the fundamental tone. It is the unique mixture of harmonics which gives each sound (or musical instrument) its distinctive timbre. The piano has a particularly complicated harmonic structure. A sample piano tone has been included in this distribution. Click on the PianoLowC icon to have GFFT analyze it for you, or click here to see a piano spectrum right now.

1.4 This is 2048 points!

Note that you were seeing the contour of 2048 spectral points, not just 32 or some such small number you would see from a toy spectrum analyzer, and that GNUPLOT (the plotting program used by GFFT) shows maps the entire contour to the resolution of the Amiga screen, so no peak or valley is missed (unlike the 'pixel averaging' approach), and you see the maximum height of each peak.

Thus, each spectral peak is a real peak which rises above the 'noise floor' that can be plainly seen at the bottom. And clearly you can count over 50 harmonics in this one note! (This is not simply because there are only 50 spectral bands.) If you were to zoom in more closely, you could determine pretty well the exact frequency of each peak (which would also show just embarassingly out of tune this piano was). You wouldn't be able to do that with any toy analyzer, and not even with many "professional" ones. (Though, I confess, some of the spectral peaks might have been enhanced by harmonic and IM distortion of my recording and sampling setup.)

1.5 Uses of Waveform Analysis

Wave analysis is also useful in various scientific and engineering work. For example, the design and quality of jet engines is evaluated acoustically. Often it is useful to know how something mechanical or biological vibrates, and that information can be found in the sound that it produces.

3D waveform analysis can give 'fingerprints' of human or animal utterances, which may be useful in determining their identity or studying their physiology or behavior. For such applications, frequently a different kind of 3-D display is used, in which amplitude is shown as a change in color. Such a display is called a 'Spectrogram,' and there is an excellent Amiga program called Amiga Spectrogram to produce these, which I highly recommend for those interested in making spectrograms. Note, however, that it will not provide anything close to the resolution of GFFT. Unfortunately, GNUPLOT does not have a spectrogram-like display mode.

1.6 3D Waveform Analysis

Since any naturally created sound will vary over time, looking at a 2-D Spectrum analysis will not give a complete picture, as it will simply show the average frequency content over the entire sample.

GFFT allows for very flexible 3-D analysis. To just get you started, however, it also provides an 'Auto 3D setup.' Click on the 3D button and then the Auto Setup button, and you are ready to perform a 3D analysis of the piano note. [The greenish parts of this display are supposed to be where you are seeing underneath the surface of the 3D contour, but some may be caused by a bug in GNUPLOT's 3D rendering algorithm.]

Basically, to do a 3D Waveform Analysis, you chop up the input sample into segments, do a spectrum analysis on each segment, and then display the results in some kind of 3D form which shows a series of spectra, changing over time.

Note in the piano 3-D analysis how the lowest harmonic starts off being the highest, then is overtaken by the middle harmonics. This partially illustrates the 'complexity' and richness of a piano's sound. In the 2D graph, you may have noticed that one of the middle harmonics was the highest, but at the beginning of the tone it is not.

1.7 Frequency Response Analysis

Frequency response analysis differs from simple wave analysis in that you must supply an input signal of some kind to the component under test, and then compare the output to the input.

GFFT has many features which improve this kind of analysis.

However, in this release of GFFT, no input signal sample is provided. You will have to provide your own. If you have a few hifi components, however, you can probably produce your own useful signal quite easily. For example, a tuner or a tape machine can be used to produce fairly good random noise. Or, better yet, if you already have you own random noise generator (as I do), and microphone (or Realistic SPL meter with phono jack output), you are set to produce the kind of graphs shown in this document. For example, this is a graph of the frequency response of my custom tri-amped modular speaker system. This may not be as 'nice' as I would like, but it is accurate.

In fact, the two curves shown plotted on top of each other are seperate samples, taken minutes apart, each having over 3 million points.* This demonstrates that the zigs and zags shown are not part of random fluctuation. I could repeat this test over and over and get almost exactly the same result.

(Note: I could make this response look flatter by changing the y axis and applying more smoothing. But why would I want to do that?)

* In fact, they were also corrected using entirely independent noise spectrum calibration files.
Correction using calibration files
will be
discussed later.

1.8 Why is Spectrum Analysis primarily used for Speakers?

Spectrum analysis is most frequently applied to transducers, such as loudspeakers and microphones. There are simpler techniques which work fairly well for most electronic equipment such as amplifiers (if you have a typical workbench of electronic instruments including an oscillator and a AC voltage meter or oscilloscope).

This is because the frequency response of purely electronic equipment such as amplifiers is fairly simple and stable under typical measurement conditions.

This is not particularly true of sound transducers, and loudspeakers in particular. Their response is difficult to measure using such simple equipment as an audio oscillator (except certain specially modified oscillators that produce sweeping or warbling tones) and a meter. Using an oscillator, you will quickly run into gigantic peaks and valleys of frequency response which are stimulated by the artificially pure and unchanging sound of the oscillator.

The measurement of loudspeakers is my particular interest, and so I will focus on that in the remainder of this document. However, many of the ideas described would apply to the analysis of other transducers.

1.9 Why not use an oscillator and a level meter?

Now, it might seem like it would be very easy to measure the frequency response of a loudspeaker system with an audio oscillator (capable of generating any frequency you can hear) and a simple loudness meter, setting the oscillator at each frequency you are interested in (say, 1/3 octave apart) and recording all the data.

This turns out not to work very well. Thanks to room resonances, even very slight turns of the frequency dial on the oscillator can result in huge differences in the loudness measurement. The pure sustained tones of an oscillator produce 'standing waves,' in an acoustic environment, which

result in the very steep peaks and valleys in the frequency response measured with a simple oscillator. During a musical reproduction, such standing waves are not usually produced to such an alarming degree.

Because pure sustained tones stimulate resonances worse than are actually perceived in music, other measurement techniques are used.

1.10 Swept tones and Maximal Length Sequences (TM)

One useful technique for measuring frequency response is to use a swept-frequency oscillator with a chart recorder. Swept-frequency oscillators can be relatively inexpensive by themselves, but they are fairly useless without a synchronized cooperating chart recorder. Such combination machines used to be sold together as 'frequency response test sets' and were quite expensive.

Nowadays, old-fashioned frequency response test sets have largely been replaced by computerized professional audio analyzers (which are still quite expensive), which may also provide other kinds of tests using specially synthesized signals with useful frequency and phase characteristics (such as the 'Maximal Length Sequences' (TM) which are popular now). These signals are formulated to so as to increase the accuracy of measurement at low frequencies while not over stressing high frequency elements, and to deal with the time delay between the generation of the signal and its arrival at the microphone.

GFFT does not yet replace computerized professional audio measurement systems. But, GFFT already may have some capabilities which are superior to such equipment, and so it can augment such equipment right now, as well as providing a baseline of capability for those who cannot afford such equipment (which typically costs about $20,000).

1.11 Why use Random Noise?

A relatively simple signal used for measuring frequency response without exaggerated resonances is random noise. Because the signal is random, there is little opportunity for standing waves to arise, and so the measurement reflects the way that the system (and room) respond to real music.

Usually, the speaker system plays noise continuously, and a spectrum analyzer of some kind analyzes the results. One advantage of this approach is that there needs to be no special coordination between the generator and the analyzer. Fairly inexpensive systems of this type are available, and they may even be built-in to fancy home equipment such as equalizers, receivers, and tape recorders.

One disadvantage is that because the noise is random, any particular short sample of it will not have predictable response. You have to average over a long period of time to get accurate results.

1.12 Why use Pink Noise?

Usually, 'pink noise' is used for acoustical measurements.

Unfiltered random noise (which is called 'white noise') concentrates most of its energy in the highest octaves. This makes it dangerous and inefficient for use in measuring audio equipment, which is generally is capable of safely generating its loudest levels at middle or lower frequencies. If you use white noise to do measurements, you will have to make measurements at lower levels, and you will have to take measurements over a longer period of time to get the same signal-to-noise ratio in your measurements, or there is danger you will burn out your high frequency transducers (tweeters).

Pink noise is a specially filtered form of ordinary random noise in which each octave has the same energy content, instead of each octave having all the energy of all the lower octaves combined, which is the case for white noise.

1.13 Why Does White Noise Has More Energy in Higher Octaves?

White noise is noise which is purely random. For example, if you had a 16-bit random number generator and connected it to a 16-bit digital to analog converter, you would get very nice white noise. Why, then, does it have this peculiar behavior that most of the energy is concentrated in higher octaves?

The answer really has to do with the nature of octaves. Each octave spans a doubling range of frequencies. For example, the range 20-40 Hz is exactly one octave, as is the range 10000-20000. But while the lower range and the higher range have the same "octave" range, they have a very different range when measured as a linear span of frequencies (there is a 20 Hz span in the first case, and a 10000 Hz span in the second), and it is exactly this linear measurement which reflects the way the energy in random noise will be distributed.

Note that each octave will have twice the frequency span as the octave which precedes it. Therefore, to modify white noise into pink noise, we need a filter which reduces the energy content of each higher octave by 1/2 (or -3dB) of the octave which precedes it. Unfortunately, the simplest filters reduce the energy content of each higher octave by some multiple of 1/4 (-6dB) of the preceding octave, so the design of a "pinking filter" is somewhat more complex than a typical high or low pass filter. It involves creating overlapping "poles" and "zeros" in such a way that they cancel each other out to produce the very gradual -3dB per octave slope over a desired range of frequencies.

1.14 How Do You Compensate for the Use of Pink Noise?

Now, although pink noise is better for measuring audio transducers for practical reasons, the spectrum analysis of a pink noise has problems similar to the creation of pink noise in the first place.

It is white noise that will appear to have a "flat" frequency response in a Fourier transformation (or relatively flat, anyway, consistent with limitations in the noise source and measurement equipment). Pink noise will appear to have response that slopes down toward the higher frequencies. This is because the frequency "bins" in the fourier transform have a width that is the same linear span of frequecies from the lowest bins to the highest bins. While these bins all have the same frequency span, their span as a musical octave will be very different. For example if the second lowest bin spans the range 20-40Hz, it spans exactly one octave, but if the highest ban spans the range 19980Hz-20000Hz, it spans a range that is a tiny fraction of an octave, since the entire octave ending at 20000Hz would be from 10000-20000Hz.

GFFT has a special "Pink" setting which compensates for the use of pink noise for most applications. It multiplies the value of each bin by a factor related to that bins "octave" width. (This is not entirely accurate for the very lowest bin which begins at 0 Hz, but that bin is not usually presented anyway.)

1.15 Why Use Large Samples?

When doing frequency response measurements with any kind of random noise, it pays to use as samples that are as large as possible. This is like making the "averaging" period on a "real time analyzer" as long as possible. Although pink noise will have an equal amount of energy in each octave or fraction thereof over the long run, in any finite period of time this will not necessarily be the case. A snapshot of any short period of pink noise will have response which is fairly random.

1.16 Show the effect of using different sample sizes

To demonstrate this effect, I've taken sample segments of varying sizes of pink noise. In each plot, I compare two non-adjacent (and therefore non-correlated) segments of the same length so you can get an idea of how variable they are. You will see that as the segments get very large the frequency content of the segments converge (as well as looking much flatter in themselves).

Size 10,000 Size 100,000
Size 1,000,000 Size 3,292,320

Details about these graphs
If you are concerned that even with the largest sample size, the response is not exactly what you thought "flat" would be, don't worry. We can compensate for that using a calibration list, as will be discussed later.

1.17 Why average?

Though we must use very large samples of pink noise to get consistent and accurate results, we must also apply a large amount of averaging. To do this, we use less than the maximum possible number of frequency bins.

The maximum possible number of frequency bins is determined by the length of the sample. For example, if we have 1024 points in our input sample (which might be 1/10 of a second at 10,240 samples/second), we can have a maximum number of half that many, or 512 bins. The number of bins must be a power of 2, and should also be equal to or less than 1/2 the number of sample points. (If it is more than that, we end up having to 'pad' the sample with zeros, which produces very spurious results. It is much better to truncate than to pad, but that is another story which is explored in gfft.help.)

The problem is, that with this number of bins, each 'bin' is subject to high degree of randomness. It is much like watching the display of a real-time spectrum analyzer (or simply a level meter) with the display response set to 'peak' or 'fast,' except it is at the theoretical limit of fast-ness.

So, much as we get nicer, more accurate and predictable response with our meter set to 'averaging,' the same is true with GFFT or any other spectrum analyzer. If what we want to do is determine the true frequency response of some sound reproducing system (as opposed to to making a 3-d map of the changes in some actual sound), and our input is random noise (which fluctuates as much as is possible), we want to do as much averaging as possible.

By choosing a smaller number of bins, we can divide up our sample into segments and do a separate analysis on each segment, and average the results. The more segments we have, the more averaging we can do, and the more reliable and accurate our results become.

But we don't want 0 bins either, since that wouldn't tell us anything about the frequency response. The more bins we have, the more frequency points we have. So, to get the very best response, we want to use a very long sample, with a reasonable large number of bins which is still much smaller than N/2 where N is the number of sample points.

For many people, 1024 is a 'reasonable' number of bins. However, I typically chose more, for more resolution, such as 4096. (I used 16,384 bins for the plots shown in this program.) Toy spectrum display programs use numbers more like 32 or 256. GFFT allows you infinite flexibility in this; you can set the number of bins to any useful number, though it is ultimately limited by the amount of memory in your computer.

1.18 Why use smoothing segments?

Although you might not have thought so, the previous plots also used a feature of GFFT known as smoothing.

Let's take a look at how plot made from the largest sample size would look without smoothing:

Pink Noise 3,292,320 Frames w/o Smoothing

For an even more startling illustration, take a look at how a plot made from 100,000 frames looks without smoothing:

Pink Noise 100,000 Frames w/o Smoothing

What is going on here? Why has the "line" become so thick at the high frequencies?

The plotting program GNUPLOT does not do any sort of 'pixel averaging.' It shows you what the graph would look like if you drew a line to each and every data point. Even if many of these data points occur within the space of one pixel on the screen. This is only fair because if you had a much larger and higher resolution screen, you might be able to see each line.

Meanwhile, at the high frequencies, the frequency band correspondings to the frequency bins stay the same size in Hz, but get smaller and smaller in terms of how much of a fraction of an octave they represent. Since pink noise has an equal amount of energy for each equal fraction of an octave, the energy in each bin is becoming smaller. (We are compensating for the high frequency 'droop' that would cause with the Pink weighting.) As the total becomes smaller, it also becomes more subject to relatively large fluctuations.

Some of this fluctuation is compensated for by having a large number of frames, in which case each bin is actually an average from the corresponding bin in many segments. If we had a large enough number of frames, (probably 1,000,000,000 or so here), the 'thickness' of the plot line at high frequencies would go away completely.

Smoothing points are "buckets" large enough to contain more than one bin within which all the X and Y values for each bin are averaged. This removes the "roughness" of the raw spectrum. (Actually, there are more sophisticated smoothing techniques which are considerably more complicated and are not currently incorporated in GFFT.)

When we chose to have smoothing points with a logarithmic X axis selected, GFFT automatically makes the 'buckets' for each smoothed point increase logarithmically with frequency. All the bins which fall in the range of each bucket are averaged, yielding an average X and Y value for each bucket. I have chosen to use 400 smoothing points which maps fairly well to the horizontal resolution of the screen (which is 640), considering the space which is taken up by the margins.

Why couldn't we simply use a smaller number of bins? In the graph here, I used 16,384 bins. Why not simply use a fairly small number, such as 256 or 512, which would also map fairly well to the horizontal resolution of the screen? The problem with that is the non-logarithmic sizing of the bins. Using FFT, each bin has a fixed width in Hz.
At the low frequecies, each bin might span an octave or more, while at the high frequecies, it would only span a small fraction of an octave. So, we would lose our resolution at low frequencies, as shown in this plot with 512 bins:

Pink Noise 3,292,320 Frames w/o Smoothing

1.19 How many smoothing segments should I use?

The choice of the number of smoothing segments to use depends on what you are trying to do. To see as much detail as possible, use more smoothing segments. To make the curve appear as simple as possible, (or maybe, as good as possible) use more smoothing segments.

For this document, I've generally used 16,385 bins, and about 200 smoothing segments. 200 smoothing segments would correspond to about 20/octave, which gives very high detail.

But, click here to see what my speaker response looks like with 3/octave.

Or, click here to see what my speaker response looks like with 1/octave.

3/octave is typically used in professional audio equipment. I think the reason is largely historical, though it has been argued for on various theoretical grounds.

Also, professional equipment typically uses a more sophisticated kind of smoothing--called 1/3 octave smoothing. This is relatively simple (though expensive) to do with analog audio circuitry. It is very complicated to do this with software. In fact, it would make the FFT portion of GFFT about three times more complicated to achieve this. Maybe some day I'll do it.

Meanwhile, it is the simple kind of smoothing that accounts for the 'jagged-ness' of spectra plots produced by GFFT, regardless of the number of smoothing segments. Neither approach is necessarily better, though many people are used to seeing the smooth curves plotted with third-octave smoothing.

1.20 What are Calibration Lists?

You will generally want to adjust your spectrum to correct for several things, including especially (1) the non-flat frequency characteristic of your random noise generator, and (2) the non-flat frequency response of your measurement microphone. The easy part about (1) is that you can measure the frequency response of your random noise generator with GFFT. It is not possible to do that for your microphone. You will have to have a calibration laboratory calibrate you microphone, or buy a calibrated microphone (with calibration curve provided) in the first place.

GFFT has a very flexible "calibration" feature to allow you to apply corrections. You can input any number of frequency response curves (in text files, one point per line) into the 'Calibration List' it uses. Some of these calibration curves can be specified in dB magnitudes (use the dB calibration gadget) while others can be specified in linear magnitudes (use the regular calibration gadget). You can use a spectrum file output by GFFT itself as calibration input.

If you enter a bad curve by mistake, you should cancel the entire list and start over.

I used the pink noise spectrum file produced by GFFT as calibration for (1). For (2), I started with the approximate calibration curve provided by Radio Shack, then fudged it _slightly_ (and within the range of nominal response) to make the results look somewhat more correct.

1.21 OK, Show me the effect of the calibration list

Here is the loudspeaker response with no correction. Notice how rapidly the response rolls off at high frequencies. (This is because the pink noise generator, sampler, and microphone all have response which rolls off rapidly at high frequencies. Meanwhile, my tweeter has a rated -3dB corner frequency close to 40 kHz.)

Here is the loudspeaker response corrected for the pink noise source (and the sampler itself--since they are tested at the same time).

(Pay no attention to position of the zero baseline in these graphs; it is arbitrary, and could easily be reset. It so happens that the baseline in the next two graphs centers around the curve itself because both the pink noise and the speaker response were sampled at the same level, and so when the pink noise calibration is applied, the curve is brought down to the baseline.)

Here is the loudspeaker response correct for both the pink noise source and the SPL Meter response. Notice that there is no particular roll-off at the highest measured frequencies.

1.22 Just how sensitive can GFFT be?

Finally, to show how sensitive GFFT can be to small changes in my loudspeaker system, I moved the tweeter backwards from its normal position by 2cm. (I have a 'modular' speaker system in which each component is in its own separate box and can be moved backwards or forwards from the others for phase adjustment.) Here I show the effect of moving the tweeter back by 2cm. The effect is just barely audible, but is clearly illustrated in the response curves layed over one another. In the 'back' position, the frequency response is a little more 'peaky' (less flat), and it sounds that way too.

Here I show the difference using 30 spectral bands.

In either display, the differences are clear and consistently reproducible. GFFT has the power to help guide you toward making improvements through small (or large) changes.

This is the end of this document for now. (Only appendices follow.) Best wishes in all your endeavors.

Charles Peterson

1.23 Appendix One: Details about Sample Size Comparison Graphs

The first 3 comparison graphs were made from analyses performed on one sample of pink noise (named noise.a1). The FRAMES and STARTFRAME commands were used to obtain uncorrelated sequences of sample frames from this file as follows:

File	STARTFRAME	FRAMES
noise.a1.1st-10000	0	10000
noise.a1.3rd-10000	20000	10000
noise.a1.1st-100000	0	100000
noise.a1.3rd-100000	200000	100000
noise.a1.1st-1000000	0	1000000
noise.a1.3rd-1000000	2000000	1000000

The last comparison graph (in which each sequence has 3,292,320 frames) was made from two separate files, which were recorded consecutively.

Other FFT Parameters: OVERLAP, HANN, BINS 4096 (10,000) or BINS 16384 (others),
SMOOTHINGSEGMENTS
200, PINK, DB, LOGX

If you are concerned about the notable high frequency roll-off, this is caused by the low-pass (anti-aliasing) filters in the sampler and by the sampling process itself. The high frequency 'ripples' may be a function of the noise generator itself.

All the frequency variation shown here can (and IS) compensated for during the speaker tests using the FFT file generated from the the last test as a calibration file, as will be shown.

1.24 Appendix Two: Trouble Displaying the Pictures

This article includes many screenshots, which are going to be displayed with a picture viewer named DISPLAY, which I know is present in AmigaDOS 2.0 and 2.1, and may exist as a link in 3.0 and greater.

Do not try to display these pictures with WDISPLAY. They will look terrible. Viewtek does a nice job, however.

If your system does not have DISPLAY, you may create a link for it in your path (for 2.0 and above) using a CLI command like this:

```
makelink c:display MYVIEWER
```

(where MYVIEWER is whatever your favorite picture viewer is. Use the complete path name for it, e.g., sys:utilities/viewtek)

OR, if you have 1.3, just try this:

```
copy MYVIEWER c:display
```

1.25 Appendix Three: Other Display Problems.

If you have started AmigaGuide indirectly, as through a file manager such as SID, you may still have trouble displaying the pictures which accompany this article. This depends on the way AmigaGuide is run by your file manager, etc. There is probably a way it can be done correctly, but, without knowing more about your file manager, etc., I wouldn't necessarily be able to help.

The safest way to show this document is by clicking on the icon provided, or to execute amigaguide from the CLI with a command like this:

```
AmigaGuide SpectrumAnalysis.guide
```