Fun with Java, How and Why Spectral Analysis Works

Baldwin explains how the Fourier transform can be used to determine the spectral content of a signal in the time domain.

Published: June 29, 2004
By Richard G. Baldwin

Java Programming, Notes # 1478

Preface

Programming in Java doesn't have to be dull and boring. In fact, it's possible to have a lot of fun while programming in Java. This lesson is one in a series that concentrates on having fun while programming in Java.

Viewing tip

You may find it useful to open another copy of this lesson in a separate browser window. That will make it easier for you to scroll back and forth among the different figures and listings while you are reading about them.

Supplementary material

I recommend that you also study the other lessons in my extensive collection of online Java tutorials. You will find those lessons published at Gamelan.com. However, as of the date of this writing, Gamelan doesn't maintain a consolidated index of my Java tutorial lessons, and sometimes they are difficult to locate there. You will find a consolidated index at www.DickBaldwin.com.

In addition, you will find some background material on DSP that is important to this lesson at http://www.dickbaldwin.com/tocdsp.htm.

Multiply-Add Operations

This lesson deals with a topic commonly know as Digital Signal Processing, *(DSP for short).*

Computational requirements for DSP

The computational requirements for implementing DSP in a computer program are usually straightforward. Almost all DSP operations consist of multiplying corresponding values contained in two numeric series and then calculating the sum of the products. Sometimes, the final sum is divided by the total number of values included in the sum. This is often referred to as a *sum-of-products* or *multiply-add* operation.

> *(This is the digital equivalent of integrating the product of two continuous functions between two limits.)*

Typical notation

Such an operation can be indicated by the symbolic notation shown in Figure 1 *(where the strange looking thing constructed of straight lines is the Greek letter sigma).*

```
           N-1
           --
           \
z =(1/N) * / x(n) * y(n)
           --
           n = 0
```

Figure 1 A typical sum-of-products operation

A sum-of-products operation

This notation means that a new value for **z** is calculated by multiplying the first **N** corresponding samples from each of two numeric series *(**x(n)** and **y(n)**)*, calculating the sum of the products, and dividing the sum by **N**.

> *(In this lesson, I will be dealing primarily with numeric series that represent samples from a continuous function taken over time. Therefore, I will often refer to the numeric series as a time series.)*

An alternative notation

The above notation requires about six lines of text to construct, and therefore could easily become scrambled during the HTML publishing process. I have invented an alternative notation that means exactly the same thing, but is less likely to be damaged during the HTML publishing process. My new notation is shown in Figure 2. You should be able to compare this notation with Figure 1 and correlate the terms in the notation to the verbal description of the operation given above.

```
z = (1/N) * S(n=0,N-1)[(x(n) * y(n)]
```

Figure 2 Alternate notation for a sum-of-products operation

This is the notation that I will use in this lesson

Preview

What is a time series?

For purposes of this lesson, suffice it to say that a time series is a set of sample values taken from a continuous function at equal increments of time over a specified time interval. For example, if you were to record the temperature in your office every minute for six hours, the set of 360 different values that you would record could be considered as a time series.

A new time series is produced

In DSP, we often multiply two time series together on a sample by sample basis. When I multiply the values in the time series **x(n)** by the corresponding values in the time series **y(n)**, that produces a new time series, which I will call **w(n)**.

Calculation of the mean or average

If I compute the sum of the individual values in the series **w(n)**, and then divide that sum by the number of samples, this is nothing more than the calculation of the mean or average value of the time series named **w(n)**. Most DSP operations boil down to nothing more complicated than calculating the average value of the product of two time series.

Knowing what to multiply, when, and why

The real trick in DSP is knowing what to multiply, when to multiply it, and why to multiply it.

Some DSP algorithms are very complex. For example, the Fast Fourier Transform algorithm, *(FFT),* involves nothing more than a lot of multiply-add operations under the control of an extremely complex algorithm.

In this lesson, I will concentrate on the Discrete Fourier Transform algorithm, *(DFT),* which is much less complex and therefore much easier to understand.

> *(While the DFT and the FFT produce the same results, the DFT typically runs much more slowly than the FFT, which is optimized for speed.)*

Multiply-add speed is important

Some DSP processes require extremely large numbers of multiply-add operations. In order to perform DSP in real time, the equipment used to perform the arithmetic must be extremely fast. That is where the special DSP chips, *(which are designed to perform multiply-add operations at an extremely high rate of speed)* earn their keep.

The net area under the curve

If you plot a time series as a curve on a graph, as shown in Figure 3, the sum of the values that make up the time series is an estimate of the net area under the curve.

> *(Assuming that the horizontal axis represents a value of zero, the sample values above the axis contribute a positive value to the net area and the sample values below the curve contribute a negative value to the net area. In the case of Figure 3, I attempted to come up with a set of sample values that would produce a net area of zero. In other words, the area above the horizontal axis was intended to perfectly balance the area below the horizontal axis.)*

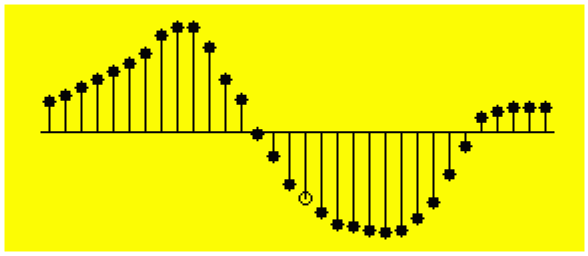

Figure 3 Plot of values in a time series

A periodic example

A periodic time series is one in which a set of sample values repeats over time, provided that you record enough samples to include one or more periods. Figure 4 shows a plot of a periodic time series. You can see that the same set of values repeats as you move from left to right on the curve plotted in Figure 4.

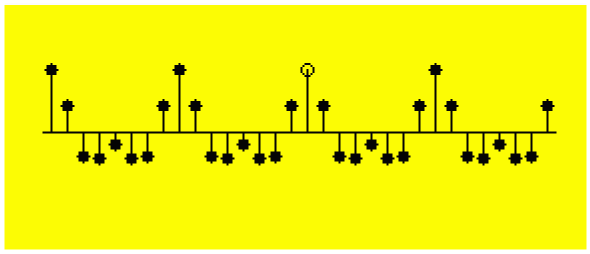

Figure 4 Area under a periodic curve

The sum of two curves

Periodic curves can often be viewed as the sum of two curves. One of the curves is the periodic component having a zero net area under the curve when measured across an even number of cycles. The other component is a constant bias offset that is added to every value of the periodic curve.

Each of the solid dark blobs in Figure 4 is a sample value. The horizontal line represents a sample value of zero. *(The empty circle is the sample value half way through the sampling interval. The only reason it is different is to mark the mid point.)*

The net area under the curve

What is the net area under the curve in Figure 4? Can you examine the curve and come up with a good estimate. As it turns out, the net area under the curve in Figure 4 is very close to zero *(at least it is as close to zero as I was able to draw it).*

Now take a look at Figure 5. What is the net area under the curve in Figure 5?

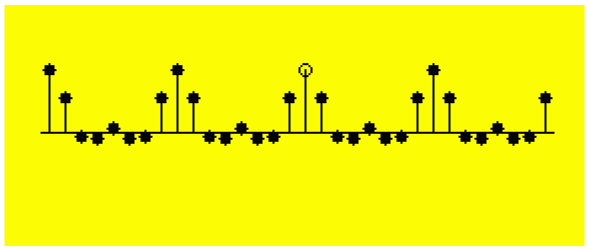

Figure 5 Area under a periodic curve with an offset

Compare Figure 5 to Figure 4

Compare Figure 5 with Figure 4. Each of these curves describes the same periodic shape *(although Figure 4 has a larger peak-to-peak amplitude, meaning simply that every value in Figure 4 has been multiplied by the same scale factor).*

However, the curve in Figure 5 is riding up on a positive bias, while the curve in Figure 4 is centered about the horizontal axis. While the net area under the curve in Figure 4 is zero, the net area under the curve in Figure 5 is a non-zero positive value.

The curve in Figure 5 can be considered to consist of the sum of two parts. One part is a straight horizontal line on the positive side of the horizontal axis. The other part is the periodic curve from Figure 4, added to that positive bias.

> *(The curve in Figure 4 can also be considered to consist of the sum of two parts. However, in Figure 4, the bias value is zero.)*

The net area

The net area contributed by the periodic part of the curve in Figure 5 continues to be zero. In effect, the non-zero net area under the curve in Figure 5 is the amount of the positive bias multiplied by the number of points. In other words, because I captured an even number of cycles of the periodic portion of the curve in my calculation of net area, only the bias contributes a non-zero value to the net area under the total curve.

Part of a cycle

Had I failed to capture an even number of cycles of the periodic portion of the curve, then the positive lobes might not completely cancel the negative lobes, and the net area under the curve would be influenced by the periodic portion of the curve in addition to the bias.

Why is this important?

These concepts are extremely important in this lesson as we learn how to do frequency spectral analysis.

As you will see in this lesson, in doing frequency spectral analysis, we will form a product between a target time series and a sinusoid. The purpose is to measure the power contained in the time series at the frequency of the sinusoid.

The product of the target time series and the sinusoid will produce the sum of a potentially infinite number of periodic functions, some of which may be riding on a positive or negative bias. We will measure the amount of bias by computing the average value of the product time series. The amount of bias will be our estimate of the power contained in the target time series at the frequency of the sinusoid.

The Fourier Transform

There is a mathematical process, known as the Fourier transform, which can be used to linearly transform information back and forth between two different domains. The information can be represented by sets of complex numbers in either or both domains.

The domains can represent a variety of different things. In DSP, the domains are often referred to as the *time domain* and the *frequency domain,* but that is not a requirement. For example, one of the domains could represent the samples that make up the pixels in a photograph and the other domain could represent something having to do with the manipulation of photographs.

Usually when dealing with the time domain and the frequency domain, the values that make up the samples in the time domain are purely real while the values that make up the samples in the frequency domain are complex.

Time domain and frequency domain

For some purposes, it is preferable to have your information in the time domain. For other purposes, it is preferable to have your information in the frequency domain. The Fourier transform allows you to transform your information back and forth between these two domains at will.

For example, it is possible to transform information from the time domain into the frequency domain, modify that information in the frequency domain, and then transform the modified information back into the time domain. This is one way to accomplish frequency filtering.

Example of time domain and frequency domain

If you were to draw a graph of the voltage impinging on the speaker coils on your stereo system over time, that would be a time series, which is a member of the time domain.

If you were to observe the lights dancing up and down on the front of your equalizer while the music is playing, you would be observing the same information presented in the frequency domain. Typically the lights on the left represent low frequencies or bass while the lights on the right side represent high frequencies or treble. Often there is a slider associated with each vertical group of lights that allows you to apply filters to emphasize certain parts of the frequency spectrum and to de-emphasize other parts of the frequency spectrum.

Forward and inverse transforms

Two very similar forms of the Fourier transform exist. The forward transform is used to transform information from the time domain into the frequency domain. The inverse transform is used to transform information from the frequency domain back to the time domain.

Sampled time series

The theoretical Fourier transform is defined using integral calculus as applied to continuous functions. As a practical matter, in the digital world, we almost never deal with continuous functions. Rather, we deal with functions that have been reduced to a series of discrete numbers *(or samples),* which are the result of some discrete measurement system.

> *(As mentioned earlier, recording the temperature in your office once each minute for twenty-four hours would produce such a discrete series of numbers.)*

Integration and summation

In many cases, the integration operation encountered in integral calculus can be approximated in the digital world by a summation operation using discrete data. That is the case with the Fourier transform. Thus, the summation form of the Fourier transform that is applied to discrete time series is known as the Discrete Fourier Transform, or DFT.

The FFT

The DFT is a computationally intense operation. Given certain restrictions involving the number of values in the time series and the number of frequencies at which the spectral analysis will be performed, there is a special algorithm that can result in computational economy. The algorithm that is used to realize that economy is commonly referred to as the *Fast Fourier Transform,* or *FFT.*

DFT versus FFT

The DFT is more general than the FFT, but the FFT is much faster than the DFT. It is important to understand that these are simply two different algorithms for doing the same thing. Either can be used to produce the same results *(but as mentioned earlier, the FFT is somewhat more restricted as to the number of time-domain and frequency-domain samples).*

Because the DFT algorithm is somewhat easier to understand than the FFT algorithm, I will concentrate on the DFT algorithm to explain how and why the Fourier transform accomplishes what it accomplishes.

The DFT

Using my alternative notation described earlier, the expressions that you must evaluate to determine the frequency spectral content of a target time series at a frequency F are shown in Figure 6 *(note that I didn't bother to divide by N).*

```
Real(F) = S(n=0,N-1)[x(n)*cos(2Pi*F*n)]
Imag(F) = S(n=0,N-1)[x(n)*sin(2Pi*F*n)]

ComplexAmplitude(F) = Real(F) - j*Imag(F)

Power(F) = Real(F)*Real(F) + Imag(F)*Imag(F)
```

Figure 6 Forward Fourier transform

What does this really mean?

Before you panic, let me explain what this means in layman's terms. Given a time series, **x(n)**, you can determine if that time series contains a cosine component or a sine component at a given frequency, **F**, by doing the following:

- Create one new time series, **cos(n)**, which is a cosine function with the frequency **F**.
- Create another new time series, **sin(n)**, which is a sine function with the frequency **F**. *(The methods needed to create the cosine and sine time series are available in the Math class in the standard Java library.)*
- Multiply **x(n)** by **cos(n)** on a sample by sample basis and compute the sum of the products. Save this value, calling it **Real(F)**. This is an estimate of the amplitude, if any, of the cosine component with the matching frequency contained in the time series **x(n)**.
- Multiply **x(n)** by **sin(n)** on a sample by sample basis and compute the sum of the products. Save this value, calling it **Imag(F)**. This is an estimate of the amplitude, if any, of the sine component with the matching frequency contained in the time series **x(n)**.
- Consider the values for **Real(F)** and **Imag(F)** to be the real and imaginary parts of a complex number.
- Consider the sum of the squares of the real and imaginary parts to represent the power at that frequency in the time series.

It's that simple

That's all there is to it. For each frequency of interest, you can use this process to compute a complex number, **Real(F) - jImag(F)**, which represents the component of that frequency in the target time series.

*(The mathematicians in the audience probably prefer to use the symbol **i** instead of the symbol **j** to represent the imaginary part. The use of **j** for this purpose comes from my electrical engineering background.)*

Similarly, you can compute the sum of the squares of the real and imaginary parts and consider that to be a measure of the power at that frequency in the time series.

(This is the value that you would likely see being displayed by one of the dancing vertical bars on the front of the equalizer on your stereo system.)

Normally we are interested in more than one frequency, so we would repeat the above procedure once for each frequency of interest.

(This would produce the set of values that you would likely see being displayed by all of the dancing vertical bars on the font of the equalizer on your stereo system.)

Why does this work?

This works because of the three trigonometric identities shown in Figure 7.

```
1. sin(a)*sin(b)=(1/2)*(cos(a-b)-cos(a+b))

  2. cos(a)*cos(b)=(1/2)*(cos(a-b)+cos(a+b))

  3. sin(a)*cos(b)=(1/2)*(sin(a+b)+sin(a-b))



  Figure 7
```

Although these identities apply to the products of sine and cosine values for single angles **a** and **b**, it is a simple matter to extend them to represent the products of time series consisting of sine and cosine functions. Such an extension is shown in Figure 8.

Products of sine and cosine functions

In each of the three cases shown in Figure 8, the function **f(n)** is a time series produced by multiplying two other time series, which are either sine functions or cosine functions.

```
1. f(n) = sin(a*n)*sin(b*n)
        = (1/2)*(cos((a-b)*n)-cos((a+b)*n))

2. f(n) = cos(a*n)*cos(b*n)
```

```
        = (1/2)*(cos((a-b)*n)+cos((a+b)*n))

3. f(n) = sin(a*n)*cos(b*n)
        = (1/2)*(sin((a+b)*n)+sin((a-b)*n))

Figure 8
```

Rewrite and simplify

Figure 9 rewrites and simplifies these three functions for the special case where **a=b**, taking into account the fact that **cos(0) =1** and **sin(0) = 0**.

```
1. f(n) = sin(a*n)*sin(a*n) = (1/2)-cos(2*a*n)/2
2. f(n) = cos(a*n)*cos(a*n) = (1/2)+cos(2*a*n)/2
3. f(n) = sin(a*n)*cos(a*n) = sin(2*a*n)/2
Figure 9
```

What can we deduce from these identities?

First you need to recall that the average of the values describing any true sinusoid is zero when the average is computed over an even number of cycles of the sinusoid.

> *(A true sinusoid does not have a bias to prevent it from being centered on the horizontal axis.)*

If a time series consists of the sum of two true sinusoids, then the average of the values describing that time series will be zero if the average is computed over an even number of cycles of both sinusoids, and very close to zero if the average is computed over a period that is not an even number of cycles for either or both sinusoids.

> *(The average will approach zero as the length of data over which the average is computed increases.)*

Product of two sine functions having the same frequency

Let's apply this knowledge to the three cases shown above for **a=b**. Consider the time series for case 1 in Figure 9. This case is the product of two sine functions having the same frequency. The result of multiplying the two sine functions is shown graphically in Figure 10.

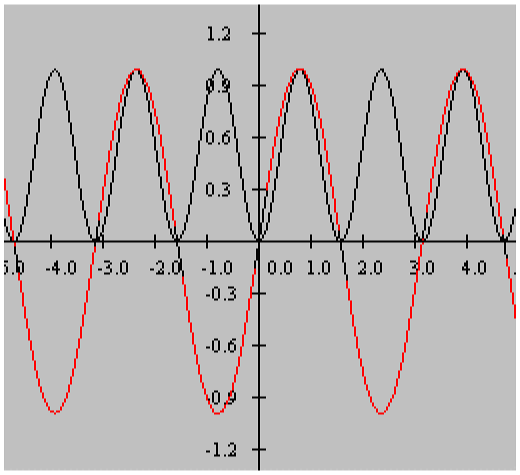

Figure 10 Plot of sin(x) and sin(x)*sin(x)

The red curve in Figure 10 shows the function **sin(x)**, and the black curve shows the function produced by multiplying **sin(x)** by **sin(x)**.

The sum of the product function is not zero

If you sum the values of the black curve over an even number of cycles, the sum will not be zero. Rather, it will be a positive, non-zero value.

Now refer back to **Imag(F)** in Figure 6. The imaginary part is computed by multiplying the time series by a sine function and computing the sum of the products. If that time series contains a sine component with the same frequency as the sine function, that component will contribute a non-zero value to the sum of products. Thus, the imaginary part of the transform at that frequency will not be zero.

Product of two cosine functions having the same frequency

Now consider the time series for case 2 in Figure 9. This case is the product of two cosine functions having the same frequency. The result of multiplying two cosine functions having the same frequency is shown graphically in Figure 11.

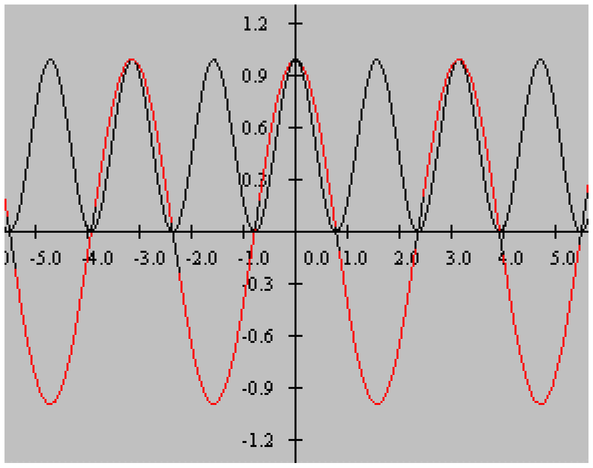

Figure 11 Plot of cos(x) and cos(x)*cos(x)

The red curve in Figure 11 shows the function **cos(x)**, and the black curve shows the function produced by multiplying **cos(x)** by **cos(x)**.

Again the sum of products is not zero

If you sum the values of the black curve in Figure 11 over an even number of cycles, the sum will not be zero. Rather, it will be a positive, non-zero value.

Now refer back to the expression for **Real(F)** in Figure 6. The real part of the transform is computed by multiplying the time series by a cosine function having a particular frequency and computing the sum of products. If that time series contains a cosine component with the same frequency as the cosine function, that component will contribute a non-zero value to the sum of products. Thus, the real part of the transform at that frequency will not be zero.

Product of a sine function and a cosine function

Now consider the time series for case 3 in Figure 9, which is the product of a sine function and a cosine function having the same frequency. The result of computing this product is shown graphically in Figure 12

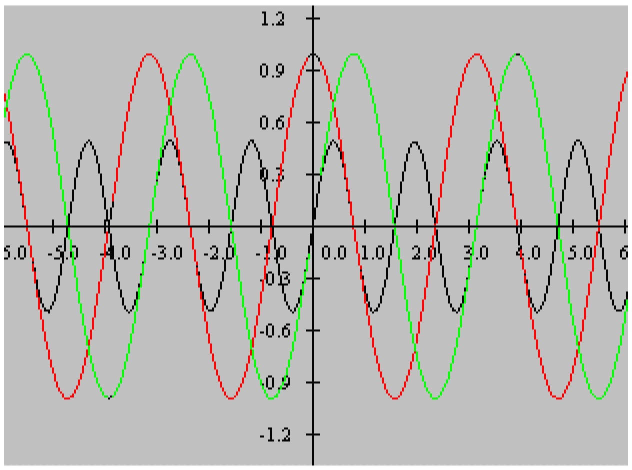

Figure 12 Plot of sin(x), cos(x), and sin(x)*cos(x)

The red curve in Figure 12 shows the function **cos(x)**, and the green curve shows the function **sin(x)**. The black curve shows the function produced by multiplying **sin(x)** by **cos(x)**.

The sum of the products will be zero

If you sum the values of the black curve over an even number of cycles, the sum will be zero.

Therefore, referring back to Figure 6, we see that the **Real(F)** computation measures only the cosine component in the time series at a particular frequency, and the **Imag(F)** computation measures only the sine component in the time series having the same frequency.

The **Real(F)** computation in Figure 6 does not produce a non-zero output due to a cosine component in the time series having the same frequency. The **Imag(F)** computation in Figure 6 does not produce a non-zero output due to a sine component in the time series having the same frequency.

Thus, at a particular frequency, the existence of a cosine component in the target time series produces the *real* output, and the existence of a sine component in the target time series produces the *imaginary* output.

Neither sine nor cosine

In reality, the sinusoidal components that make up a time series will not usually be sine functions or cosine functions. Rather, they will be sinusoidal components having the same shape as a sine or cosine, but not having the same value at zero as either a sine function or a cosine function. However, it can be shown that a general sinusoidal function can always be represented by the sum of a sine function and a cosine function having different amplitudes.

> *(A proof of the above statement is beyond the scope of this lesson. You will simply have to accept on faith that a general time series can be represented as the sum of a potentially infinite number of sine functions and cosine functions of*

different frequencies and different amplitudes. It is these cosine and sine functions that constitute the real and imaginary components of the complex frequency spectrum.)

What about non-matching frequency components?

Now we know what happens when the frequency of the **sin** and **cos** terms in Figure 6 match the frequency of sine and cosine components in the target time series. Another important question is, what do sine and cosine components in the target time series with frequencies different from the **cos** and **sin** terms in Figure 6 contribute to the output?

The answer is not very much. Referring once more to the functional forms produced by multiplying sine and cosine functions *(repeated in Figure 13 for convenience)*, we see that for any values of **a** and **b**, where **a** is not equal to **b**, the function produced by multiplying two sinusoids will be the sum of two other sinusoids. The sum of the values for any sinusoid computed over an even number of cycles of the sinusoid will always be zero, and these sinusoids are no exception to that rule.

```
1. f(n) = sin(a*n)*sin(b*n)
        = (1/2)*(cos((a-b)*n)-cos((a+b)*n))

2. f(n) = cos(a*n)*cos(b*n)
        = (1/2)*(cos((a-b)*n)+cos((a+b)*n))

3. f(n) = sin(a*n)*cos(b*n)
        =(1/2)*(sin((a+b)*n)+sin((a-b)*n))

Figure 13
```

Sum and difference frequencies

For any pair of arbitrary values for **a** and **b**, the frequencies of the sinusoids in the resulting functions will be given by **a+b** and **a-b**.

*(These are often referred to as the sum and difference frequencies. Note that as **a** approaches **b**, the difference frequency approaches zero. This is what produces the constant values of 1/2 in Figure 9 and the positive bias on the black curve in Figures 10 and 11.)*

A form of measurement error

As a practical matter, if those resulting sum and difference frequencies are not multiples of one another, it will not be possible to perform the summation over an even number of cycles of both sinusoids. Therefore, one or the other, or perhaps both, will contribute a small amount to the sum of products due to including a partial cycle in the summation. This is a form of measurement error that occurs when performing frequency spectrum analysis using Fourier transform methods.

(The percentage contribution of this error to the average value decreases as the number of cycles included in the average increases.)

Product of two sine functions at different frequencies

Consider the product of two sine functions at different frequencies as shown in Figure 14. This is a plot of the function produced by multiplying **sin(1.8x)** by **sin(2.2x)**.

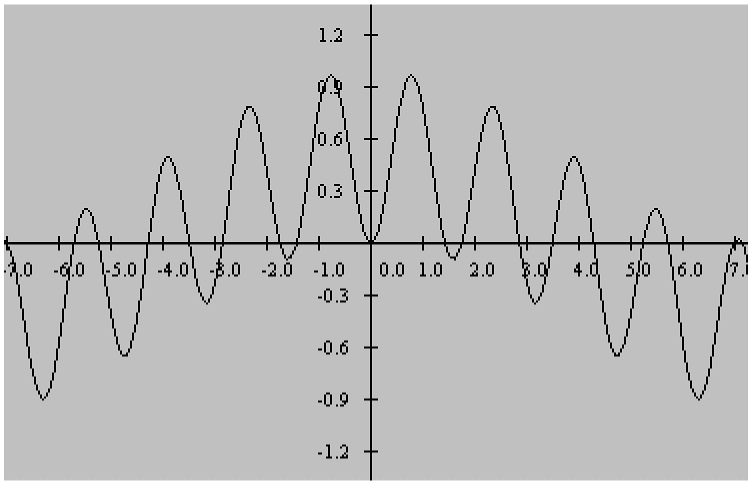

Figure 14 Plot of sin(1.8x)*sin(2.2x)

The high frequency component shown in Figure 14 is the component attributable to **a+b**. The long sweeping low frequency component is the component attributable to **a-b**.

The sum of the product function

Judging from the graph in Figure 14, performing a summation of the product function from -7 to +7 would include almost exactly nine cycles of the high frequency component. Thus, the high frequency component would contribute very little, if anything, to the summation.

However, the summation would include less than one complete cycle of the low frequency component. Therefore, the low frequency component would contribute some output to the summation in the form of a measurement error.

Similar results occur when multiplying a cosine function by a cosine function having a different frequency, or when multiplying a cosine function by a sine function having a different frequency. Graphical results for these two cases are shown in Figures 15 and 16.

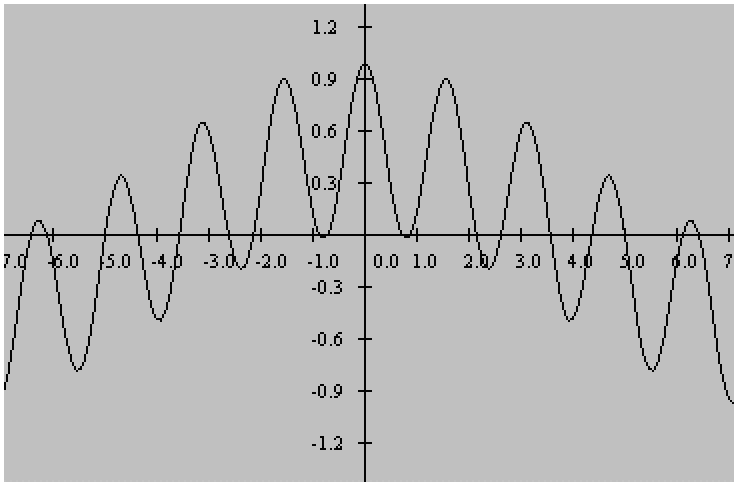

Figure 15 Plot of cos(1.8x)*cos(2.2x)

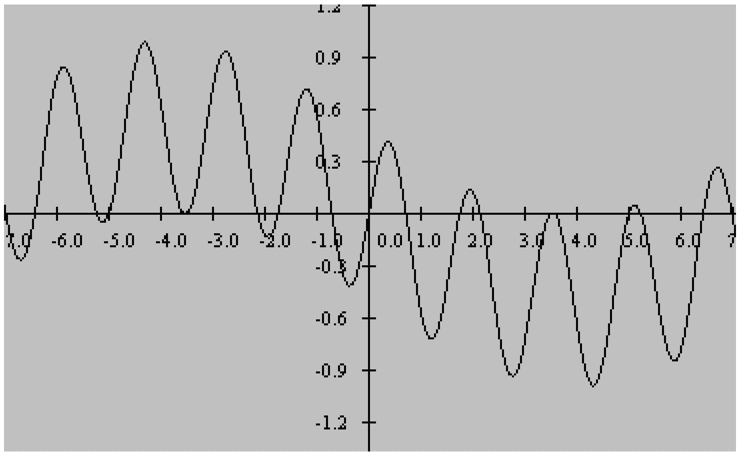

Figure 16 Plot of sin(1.8x)*cos(2.2x)

Once again, unless the summation interval includes an exact number of samples of both the sum frequency and the difference frequency, one or both of the sinusoids contained in the product function would contribute a measurement error to the result.

Summary

In this lesson, I have provided a quasi-theoretical basis for frequency spectrum analysis.

A pure theoretical basis for frequency spectrum analysis involves some rather complicated mathematics and is somewhat difficult to understand. However, from a practical viewpoint, it is not too difficult to understand how the complex mathematics produce the results that they produce.

Hopefully the quasi-theoretical explanations provided in this lesson will help you to understand what makes spectrum analysis work.

What's Next?

The next lesson in this series will reduce much of what I have discussed in this lesson to practice. I will present and explain a program that implements a DFT algorithm for performing frequency spectrum analysis. In addition, I will present the results of several interesting experiments in frequency spectrum analysis using that algorithm.

A subsequent lesson will explain some of the signal processing concepts that made it possible for the inventors of the FFT algorithm to design a computational algorithm that is much faster than the DFT algorithm. As is often the case, however, the FFT algorithm trades off speed for generality.

About the author

Richard Baldwin *is a college professor (at Austin Community College in Austin, Texas) and private consultant whose primary focus is a combination of Java, C#, and XML. In addition to the many platform and/or language independent benefits of Java and C# applications, he believes that a combination of Java, C#, and XML will become the primary driving force in the delivery of structured information on the Web.*

Richard has participated in numerous consulting projects, and he frequently provides onsite training at the high-tech companies located in and around Austin, Texas. He is the author of Baldwin's Programming Tutorials, which has gained a worldwide following among experienced and aspiring programmers. He has also published articles in JavaPro magazine.

Richard holds an MSEE degree from Southern Methodist University and has many years of experience in the application of computer technology to real-world problems.

baldwin@DickBaldwin.com

-end-