The Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT)

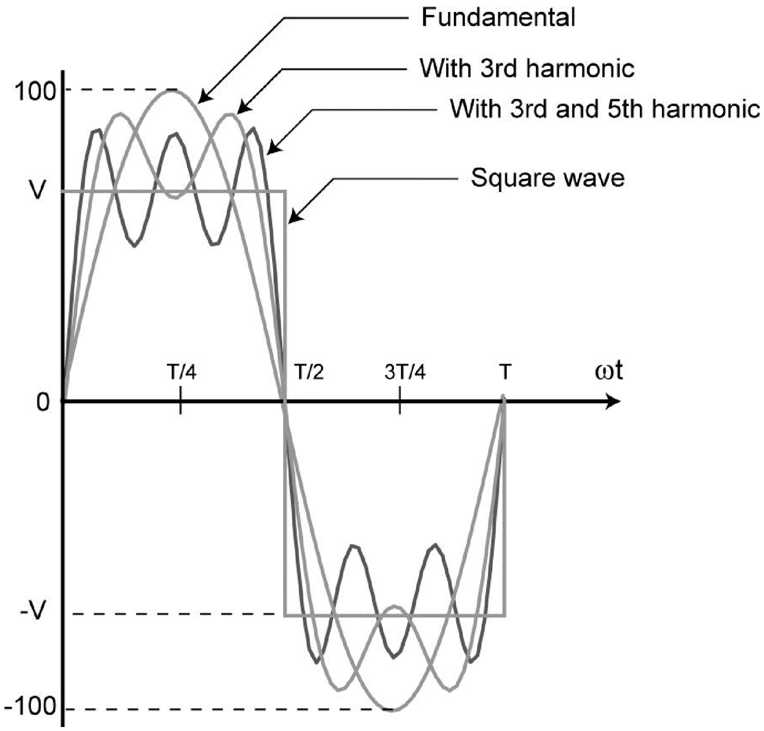
Determining Frequency Content

The second most widely used application of DSP is to determine the spectral content of a complex signal. That is, the DSP performs calculations on a sampled signal such that it can determine the frequency content of a signal. This technique is based upon the Fourier theory which states that any complex waveform is made up of sine and/or cosine waves of harmonically related frequencies with different amplitude and phases.

DSP can provide the equivalent of a spectrum analyzer output. Its output is a sequence of sample values that state the relative amplitudes of the fundamental sine wave and all harmonics. Phase plots can also be developed.

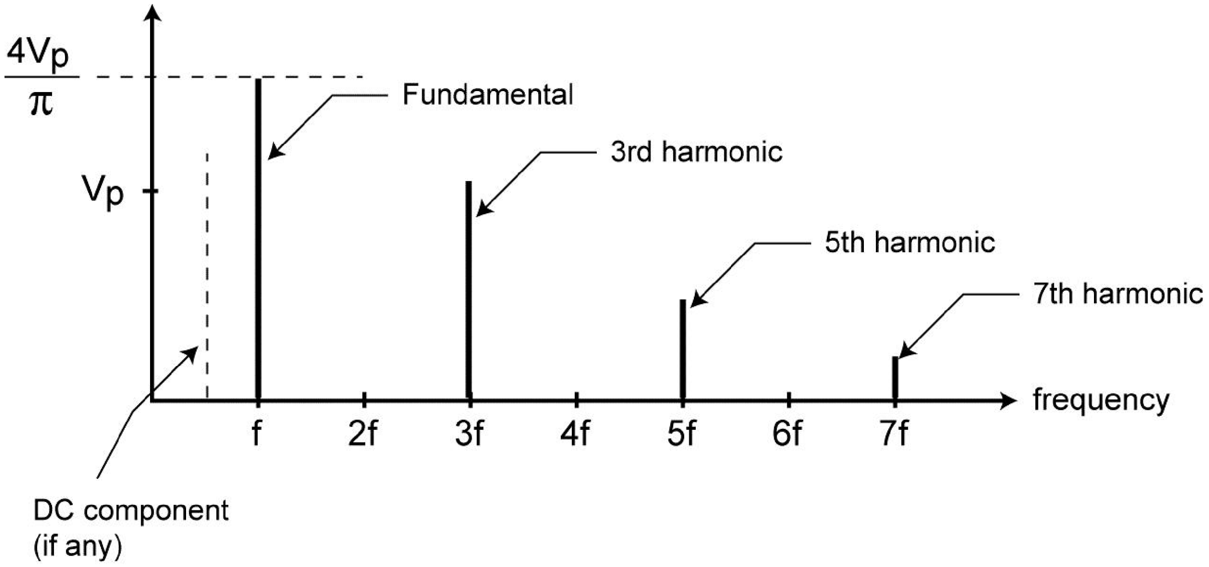
The mathematical algorithms for doing this frequency analysis are the discrete Fourier transform (DFT) and the fast Fourier transform (FFT).

Fourier Square Wave



A square wave can be synthesized by adding together a fundamental sine wave and all its odd harmonics (3rd, 5th, 7th, etc.) as shown above.

Fourier Example



If you start with an AC square wave of 1 MHz and connect it to a spectrum analyzer, the frequency domain output displayed on the CRT will be a plot of the frequency spectrum. As you can see, the plot shows a fundamental sine wave at 1 MHz and progressively smaller harmonic sine waves at 3 MHz, 5 MHz, 7 MHz and so on.

The Discrete Fourier Transform (DFT)

The DFT is simply a form of the more common Fourier analysis that engineers use to analyze complex waveforms. Using integral calculus, a continuously repetitive signal is divided up into math expressions showing the sine and cosine wave content at each frequency. A version of the Fourier series called the Fourier transform is used analyze aperiodic signals. Aperiodic signals are signals like pulses that occur at random or do not reoccur on a regular basis.

The DFT is simply a version of Fourier analysis but using sampled signals. The inputs are the periodic samples that come from an ADC and stored in a memory. The DFT algorithm is applied and the output is a series of new samples that indicate the amplitude and phase angle of each sine and cosine wave that makes up the complex wave being analyzed.

The DFT Algorithm

Here is what the DFT algorithm looks like in its raw mathematical form.

$$X(k) = (1/N) \sum x(n)e^{-j2\pi nk/N}$$ for values of n from 0 to N-1.

In this ugly and daunting equation, X(k) are the spectral output values representing the amplitudes and phases of the different sine and cosine waves. The symbol ∑ simply means the summation of multiple values of the signal samples represented by the term x(n). Each sample x(n) is multiplied by $e^{-j2\pi nk/N}$ where n and k are the numbers representing the different samples and spectral points and N is the total number of samples and spectral points. The value e is the familiar constant 2.71818 while j is the square root of -1 ($\sqrt{-1}$), the imaginary operator.

The DFT Algorithm Simplified

There is an identity in calculus called Euler's rule that says that you can replace the term $x(n)e^{-j2\pi nk/N}$ with the equivalent expression [cos(2πnk/N) – j[sin2πnk/N]. Doing this gives us a new version of the DFT that looks like this:

$X(k) = (1/N) \sum x(n) [\cos(2\pi nk/N) - j[\sin 2\pi nk/N]$ for multiple values for k and n 0 through N.

Now looking at this new expression, you begin to see how this might produce a frequency analysis of a wave. What the expression says is that we multiple each input sample x(n) by a mathematical expression for a sine wave of different frequencies and cosine wave of different frequencies and then sum the results.

What this process does is literally to compare the input wave with sines and cosines to see how closely they resemble one another. Then on the basis of how well they compare we can tell if that sine or cosine wave frequency is contained within the wave. That is a bit of an oversimplification but it is essentially what this algorithm does.

Correlation

Correlation is the mathematical process of determining the similarity between sampled two signals. What we do is multiply each sample of one signal by each sample of the other signal, add them up and plot the output samples. Looking at the output samples, we can tell the degree of similarity between the two.

The correlation process is described mathematically by the expression similar to the one we used earlier for digital filters.

$y(n) = \sum h(k)\, x[n - k]$ where k varies from 0 to some selected maximum value. Here k is the number of samples in the window selected for the calculation.

Correlation Example

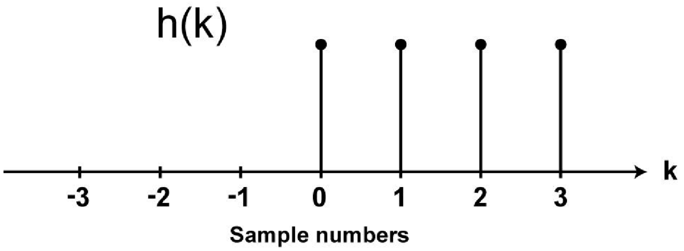
Instead of a sample x being multiplied by a coefficient, we have two sets of samples h and x multiplied by one another. Expanding the equation, we get:

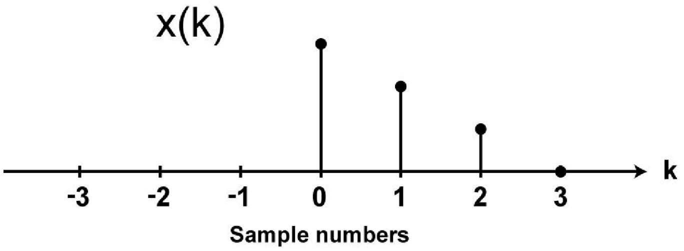
$y(n) = h(k)\, x[n] + h(k)\, x[n - 1] + h(k)\, x[n - 2] + h(k)\, x[n - 3] + \ldots\ldots$

What this represents is multiplying the first h sample by every x sample then summing. That gives only one output sample that represents a point of the output correlation plot. The process continues by shifting one set of samples over one increment and repeating the process to get a second point on an output sequence. One set of samples is then again shifted over one increment and the process is repeated. This goes on until all samples of one input are multiplied by all samples of the other input.

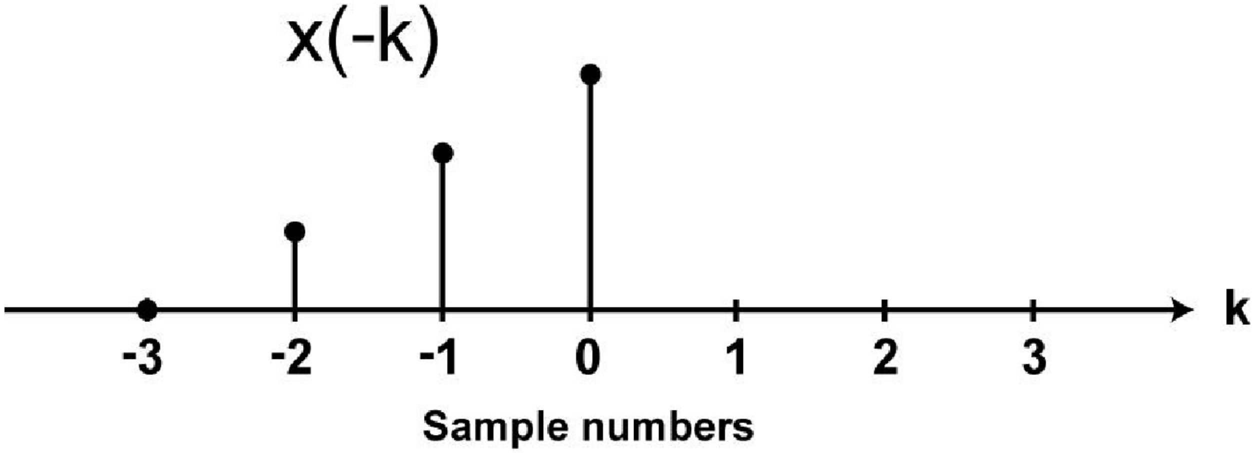
Convolution

Convolution is an operation that is similar to correlation in that we multiply samples from one input signal by the samples of another signal but we invert or flip the values of one group of samples in time. That means we multiply one set of samples by the samples of the other in the reverse order. The figures show an input sequences, h(k), which is a sampled rectangular pulse and a sampled negative-going sawtooth wave x(k).





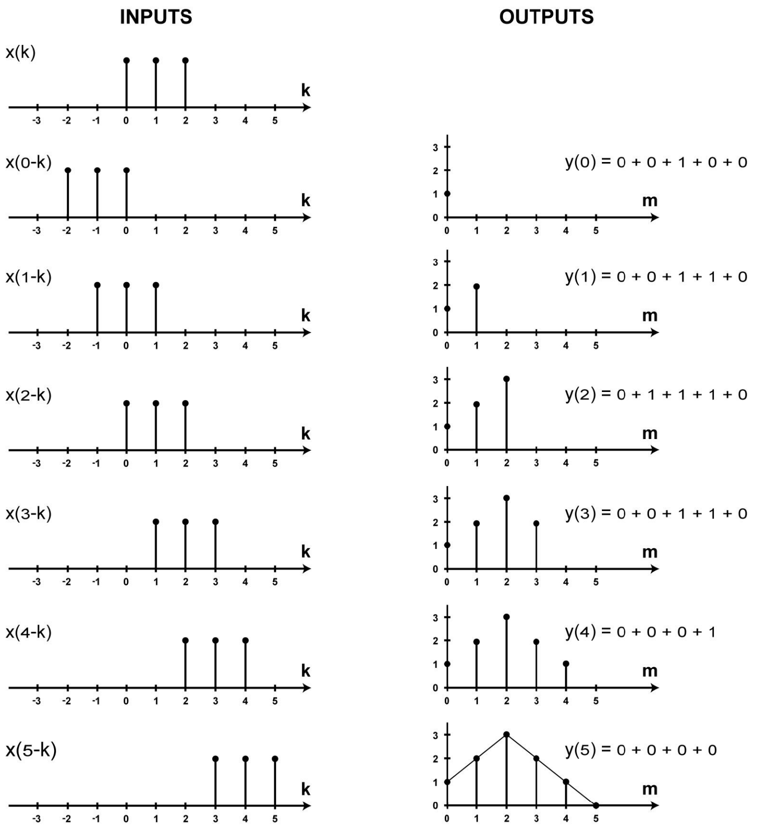
Convolution Example



Before the correlation calculation is performed, the x values are reversed as shown above. It is as if the x samples have been rotated around the 0 sample value. This process is called flipping. What this does is to align the sample values to simplify the calculation in the computer. The correlation result is still the same.

Illustrating Convolution

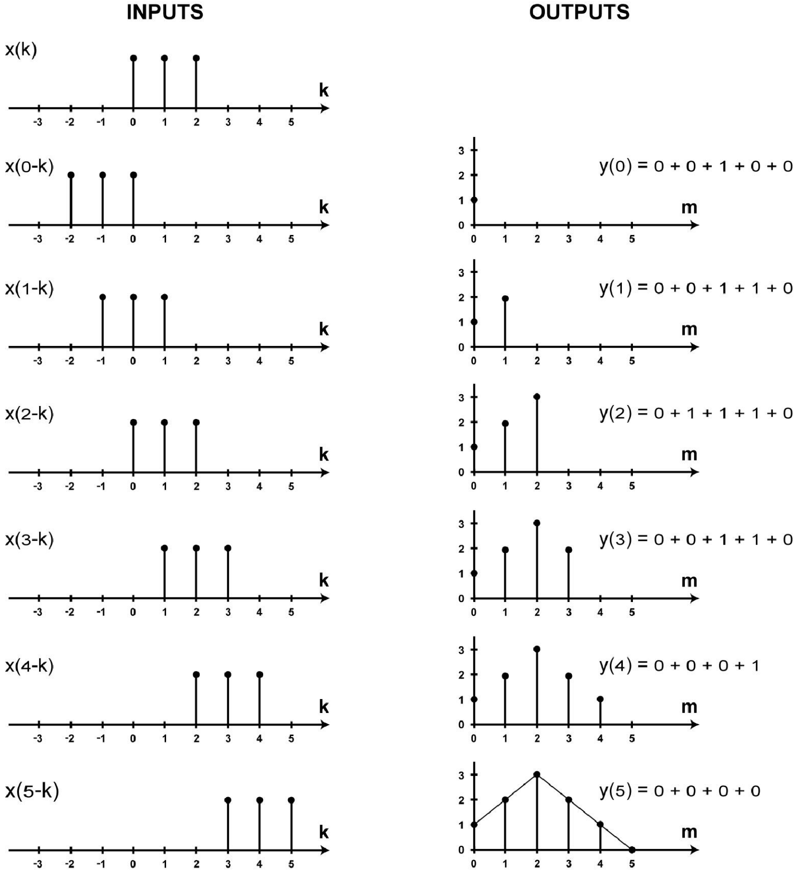
The two sampled signals are rectangular pulses with a fixed amplitude of 1 as shown in the example. The two pulses are aligned so that sample 0 of the x input and 0 sample of the h input are aligned. Note that the x samples have been flipped. All samples of x and h are then multiplied and summed producing the first point on the y output designated y(0).



NOTE: Each sample value = 1

Convolution Example

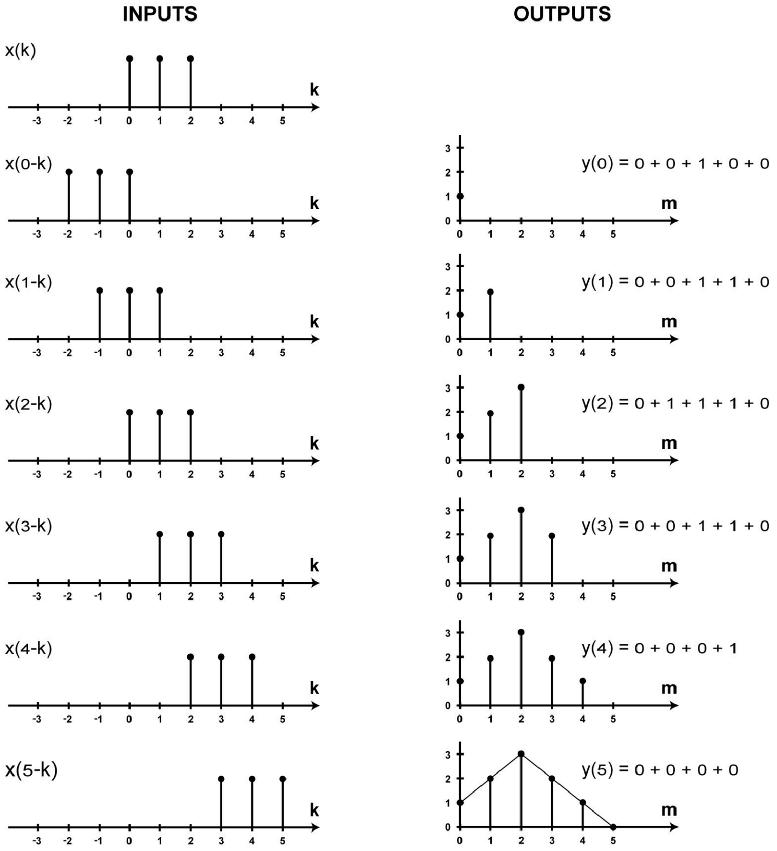
The x samples are shifted over to the right one increment then the multiply and add process is repeated. This produces output y(1). This procedure of shift-multiply-add is followed again until all samples have been multiplied by all other samples.



NOTE: Each sample value = 1

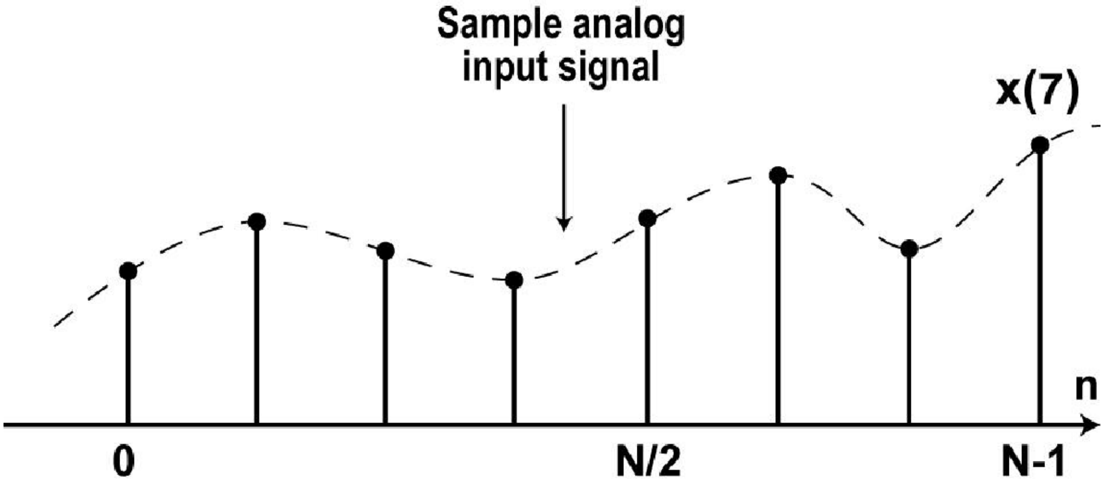
Correlation Plot

All the samples with 0 values produce 0 outputs so the process is somewhat simplified. The resulting correlation output plot is shown on the bottom right. This triangular shaped output indicates perfect correlation between the two signals since they are the same. Using two different sampled signals results in a different shaped output and a lower percentage of correlation. This is the process used in computing the DFT.



NOTE: Each sample value = 1

Implementing the DFT: Time Domain

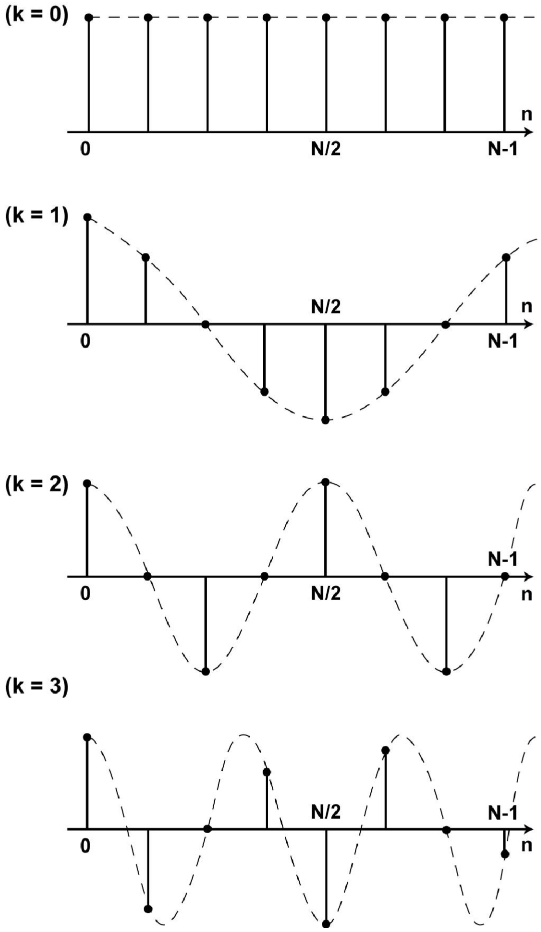


N = 8

This figure and the figures on the following pages show a simplified example of how the DFT decomposes a sampled analog wave into a frequency spectrum output. A sampled analog wave is shown above. The calculation window is N = 8 samples. This input is then compared to each of four other signals called basis functions in sampled form using the convolution process.

Implementing the DFT: Basis Function

The upper basis form is what a DC level would look like as a sampled wave. Below that is one cycle of a cosine wave representing the fundamental frequency. The third wave down in is actually two cycles of a cosine wave representing the 2nd harmonic. The fourth wave down is three cycles of a cosine wave representing the 3rd harmonic.

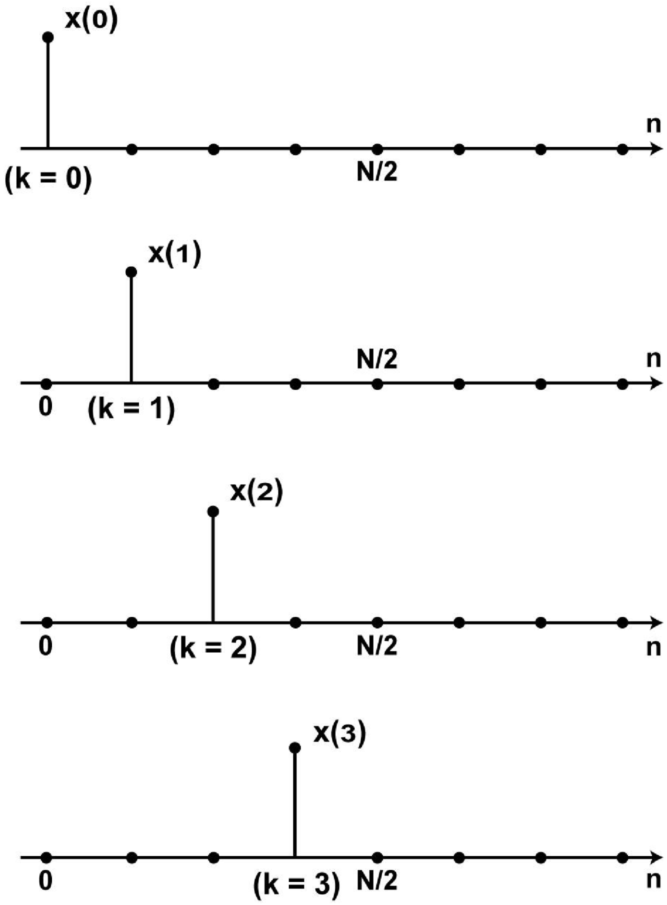


WORKREADY ELECTRONICS

Implementing the DFT: Frequency Domain

The DFT is performed by implementing a convolution between the sampled input and each basis function. Each sample in the input wave is multiplied by each sample in the basis function then one is shifted and the process is repeated until all samples have been multiplied.

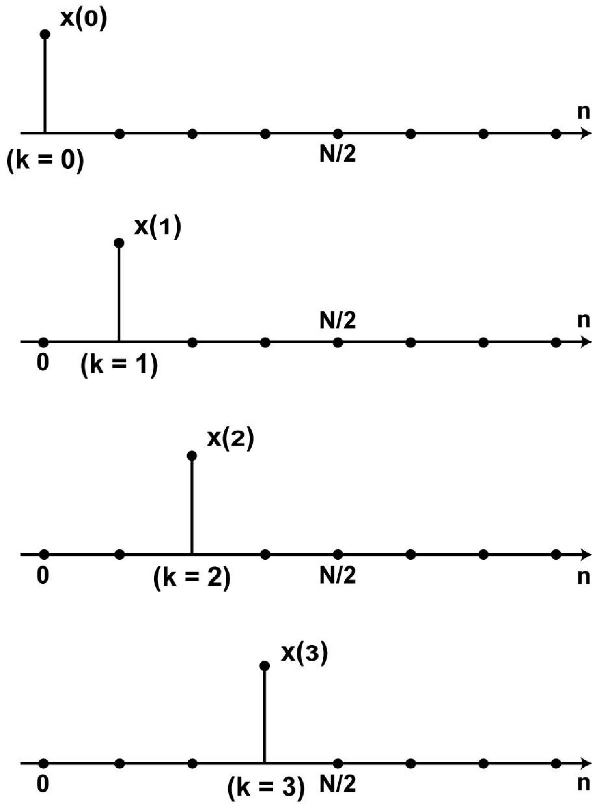
The process is repeated on the other basis functions. The results for each correlation X(n) is given here. The convolution results show the relative amplitudes of the DC average X(0) as well as the fundamental, 2nd, and 3rd harmonics.



X Outputs

The X outputs are frequency domain plots. The upper plot shows a vertical output at k=0 which indicates that the input signal has an average DC component. This is obvious from the sampled analog signal input since all samples are above the horizontal axis.

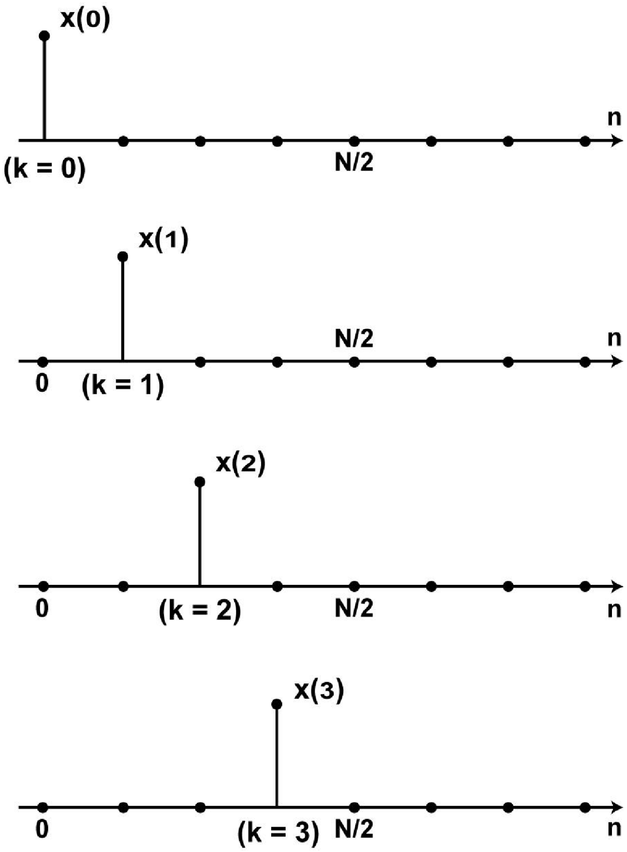
The second output X(1) shows a vertical line at k=1. This indicates that the input signal does contain the fundamental frequency cosine wave. The other outputs X(2) and X(3) show outputs of the 2nd and 3rd harmonics.



Inputs

If the sampled input was just a cosine wave at the fundamental, there would only be an output at k=1. There would be NO spectral indications at k=0, k=2 or k=3.

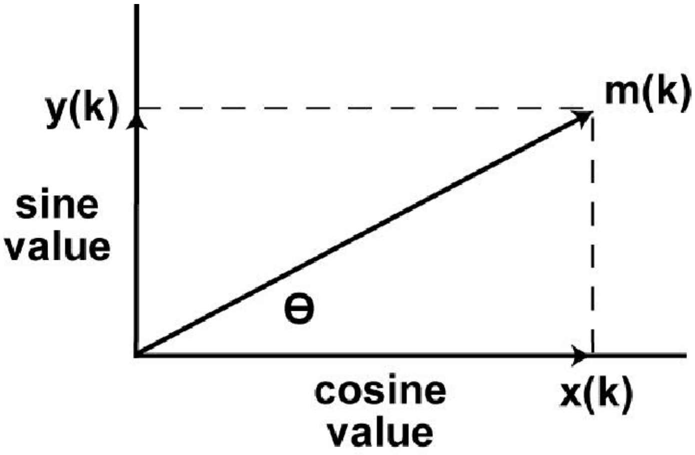
If the input was a sampled sine wave at the fundamental frequency, there would be no output even at k=1. Correlating a sine wave and a cosine wave produces zero correlation.



Analysis Continues

The analysis continues by now correlating the sampled input wave with sine wave basis functions at the fundamental and harmonic frequencies. A similar output frequency plot is obtained but with different values.

We end up with two frequency domain spectrum plots: one of cosine waves with X values and another of sine waves with Y sample values. To ensure only one composite output frequency plot, we combine the output values.

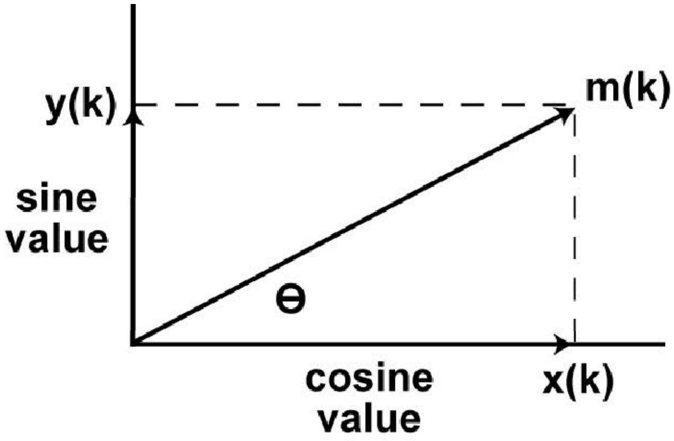


$$m \quad = \sqrt{x(k)^2 + y(k)^2}$$

$$\Theta(k) \quad = \tan^{-1}\left(\frac{y(k)}{x(k)}\right)$$

Spectrum Plot

The cosine value is shown on the horizontal axis of a right triangle and the sine value is shown on the vertical axis. Standard Pythagorean math is performed to calculate the length of the hypotenuse which is the composite magnitude of the sine and cosine values (M). This is the value displayed in the final output spectrum plot.



$$m = \sqrt{x(k)^2 + y(k)^2}$$

$$\Theta(k) = \tan^{-1}\left(\frac{y(k)}{x(k)}\right)$$

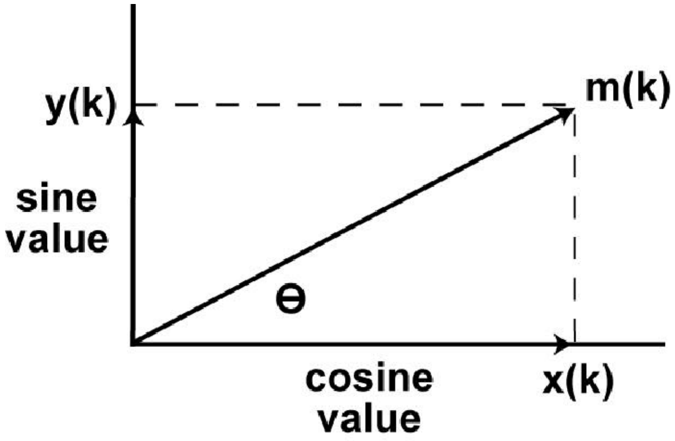
Spectrum Plot: Phase Angle

With both sine and cosine samples available, the phase angle can also be computed producing a separate phase output plot. Remember the familiar expression:

Phase angle θ(k) =

$\tan^{-1}[Y(k)/X(k)]$

These values can actually be shown in a plot on a computer screen that resembles those shown on a spectrum analyzer screen. Otherwise, the values calculated remain in memory to be used by other processes.



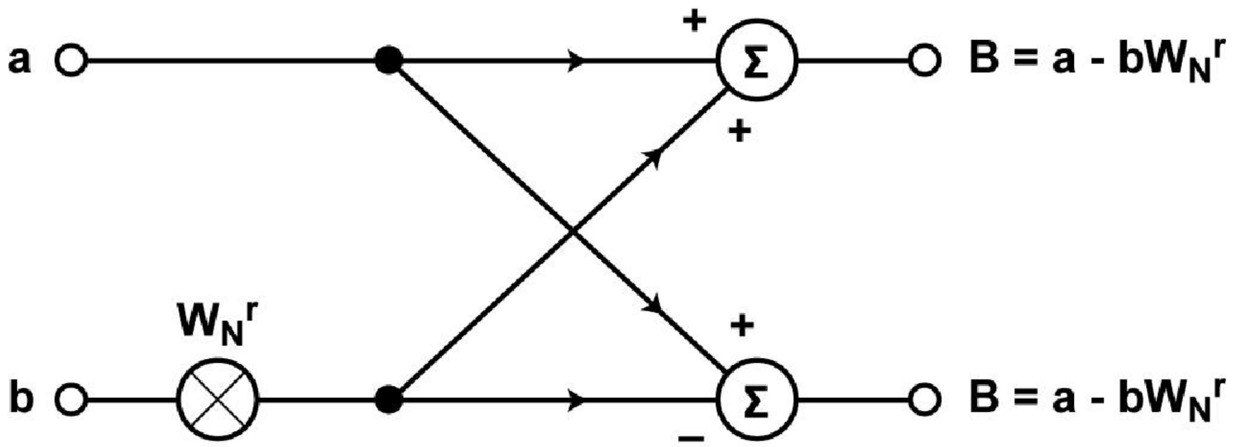
$$m = \sqrt{x(k)^2 + y(k)^2}$$

$$\theta(k) = \tan^{-1}\left(\frac{y(k)}{x(k)}\right)$$

The Fast Fourier Transform (FFT)

Looking back at the DFT example, you can imagine how much processing must be done to get the final result. The sampled input is correlated with each sine and cosine basis function and the final plots computed. It takes a great deal of processing time to analyze a signal. The greater the number of sampling points (N) the more precise the calculation but the longer the processing time. The number of multiplications that must be implemented is N^2. For example, if we use a window on the sampled input with 256 samples points then the DFT takes $256^2 = 65{,}536$ multiplications. Since multiplication in a microcomputer takes considerable time, the DFT is often too slow. The higher the frequency of the signal the faster the computation must be. This makes the DFT unusable at the higher frequencies simply because the processors are just not fast enough. This problem is solved with the FFT.

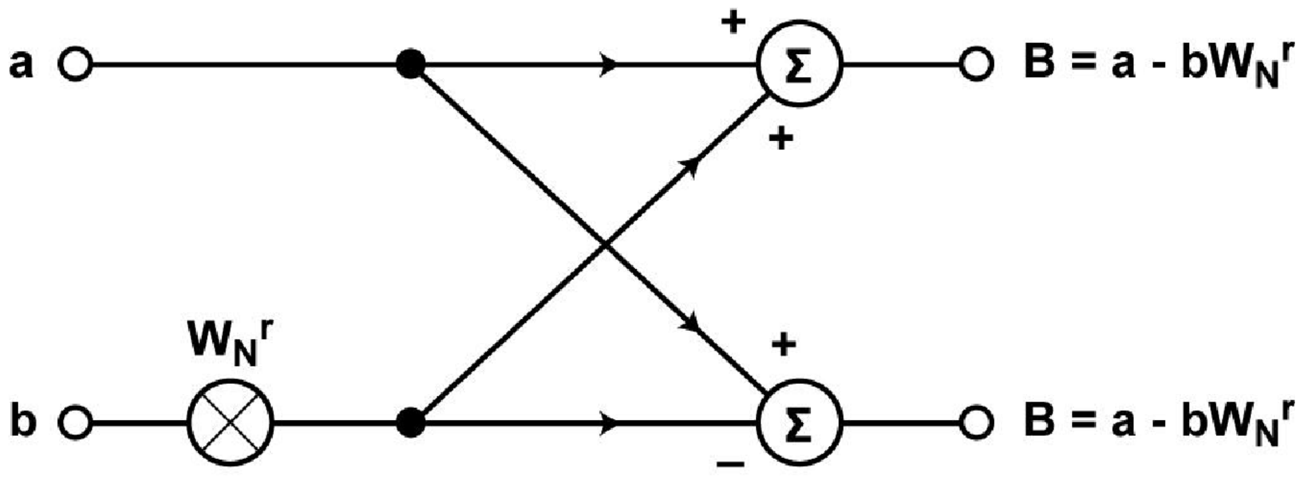
Implementing the FFT



The FFT is simply a faster way to perform a DFT. Instead of N^2 multiplications, the FFT takes $(N/2)\log_2(N)$. For a 256 point analysis, instead of 65,536 multiplications, only 1024 are required with the FFT. This is a huge improvement in efficiency resulting in far less computing time. The FFT makes the DFT process practical because calculations can be performed between samples.

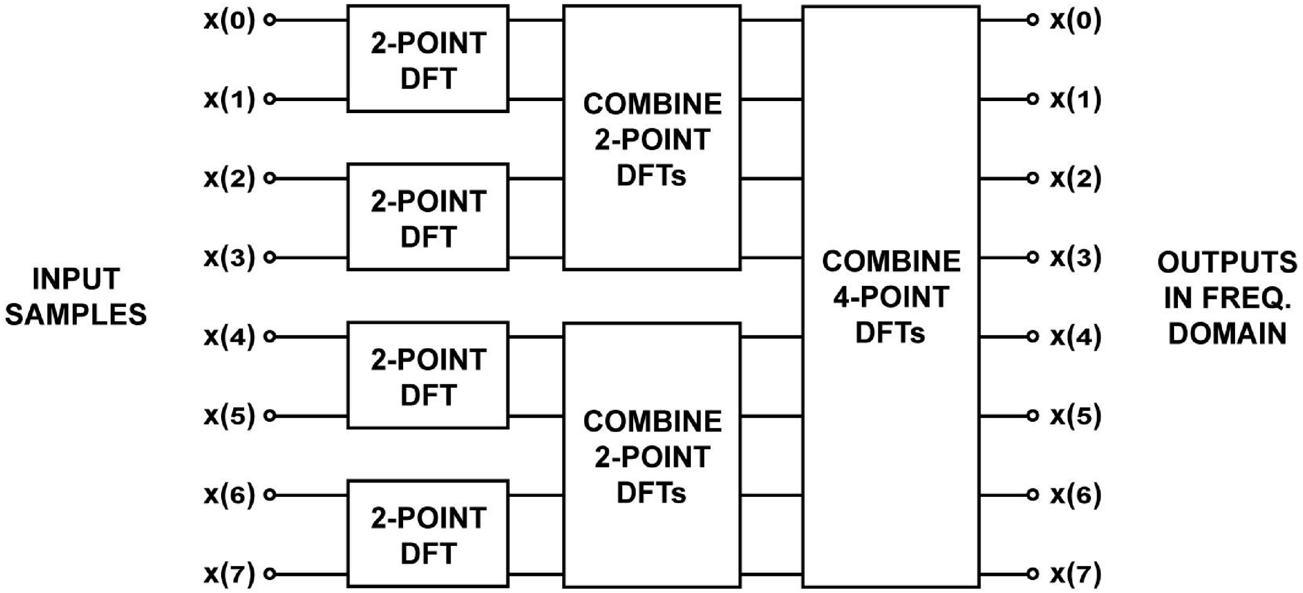
The FFT breaks up the computation into smaller faster segments. The basic computing algorithm is shown in the figure.

Butterfly Computation



The basic computing algorithm is called a two point DFT. The inputs a and b are signal sample values while the mysterious W_{Nr} values are coefficients called twiddle factors that essentially represent the basis factor values. Note the cross over multiplication, the inversion of the b factor, and summation that occurs. Because of the shape of this diagram, it is referred to as a butterfly computation. A and B values are the frequency domain outputs.

Final Output

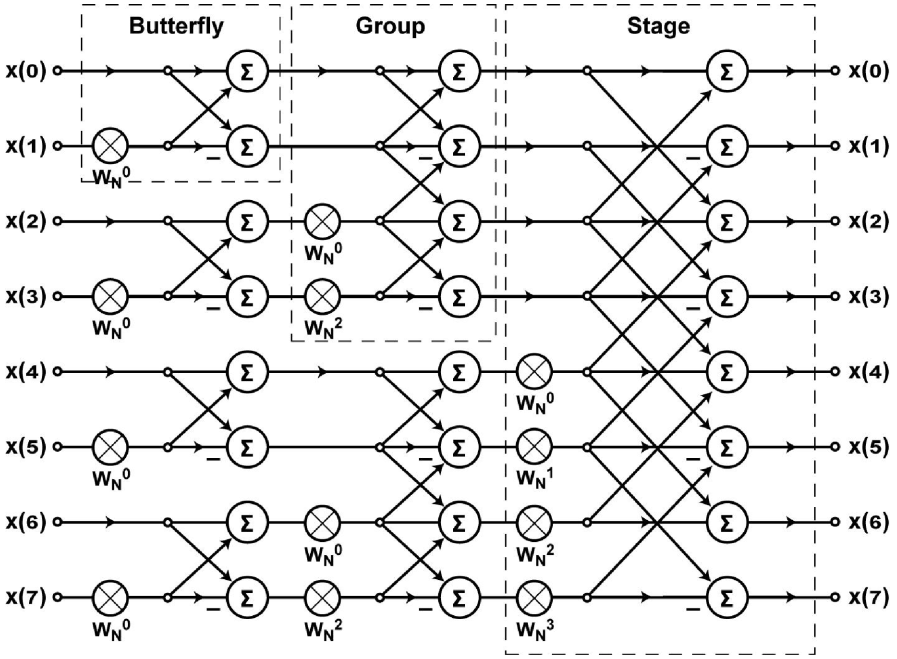


The butterfly computations are then combined to produce the final output. The figure shows how an eight point FFT is implemented. The two point DFTs are combined into four point outputs which are then combined into eight outputs.

Final Output: 8 Point FFT

This figure shows the entire process for an eight point FFT. Most FFTs use many more points to improve accuracy and resolution. The number of points used is some power of 2.

Because of the efficiency of the FFT and the super speed of modern DSP chips, it is possible to do real time spectrum analysis on even high frequency signals.



Test your knowledge

Digital Signal Processing
Knowledge Probe 3
The Discrete Fourier Transform and The Fast Fourier Transform

Click on Course Materials at the top of the page.

Then choose **Knowledge Probe 3**.