Fourier Transform for Dummies

What is a “Fourier Transform” ?

A Fourier Transform is a method of detecting what frequencies are present in a sampled waveform. Given a set of sample data and a frequency the transform will give you the amplitude and phase of that frequency within the sample data.

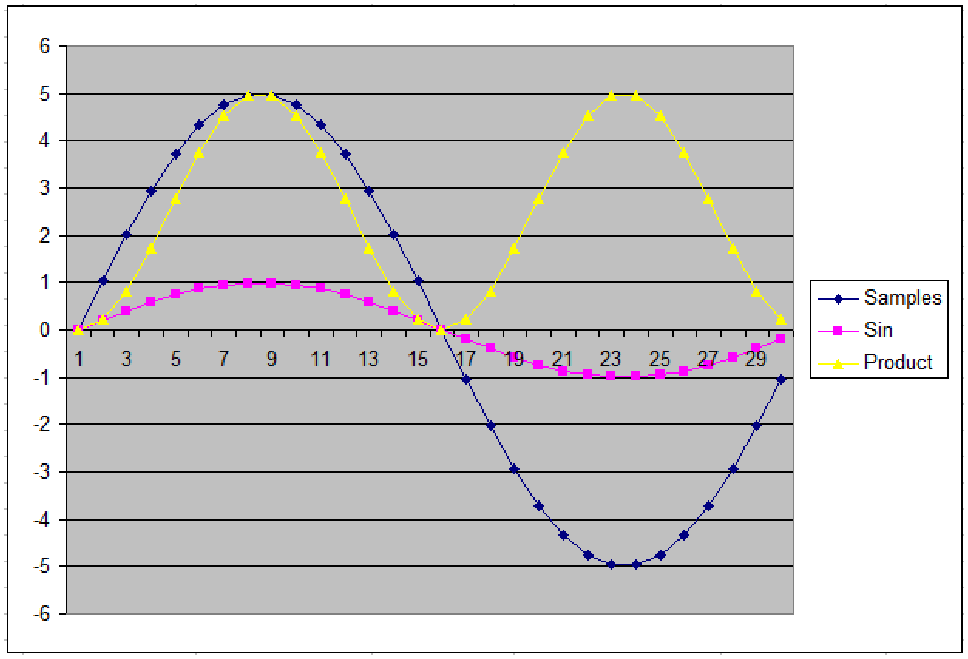
For example, say you want to know if 1Khz is present in your sample data. The Fourier transform will tell you the amplitude and phase of any 1Khz component in your sample data.

How does it work ?

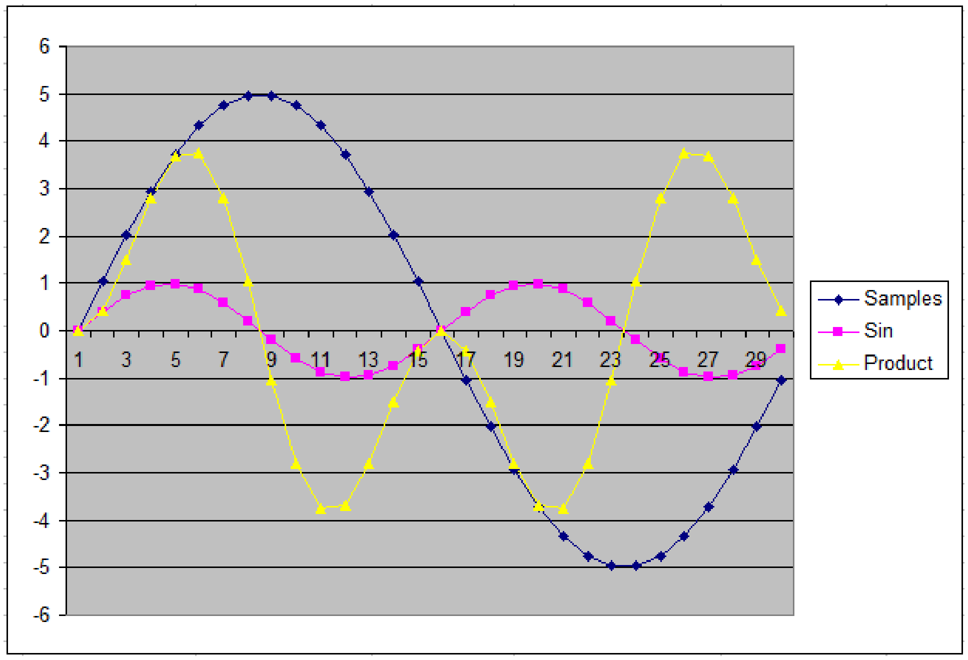
There are different types of Fourier transforms. Here we will look at the simple “Discrete Fourier Transform” or DFT.

Let’s look at a simple ideal example. Let’s say you have a 5 volt, 1Khz sine wave and you sample it at 30Khz for 1 millisecond. You would get 30 data points showing exactly 1 complete waveform. For now, let’s assume you started sampling at 0 degrees. (Blue on graph).

Now if you want to detect 1KHz, you would multiply each data point by a 1 volt, 1Khz sine wave [Magenta on graph]. The two waves are the same shape, so when they both go positive you will get a positive product (result of multiplication)[Yellow on graph], and when they both go negative you will also get a positive product (negative times a negative is positive). Now if we take the sum of the 30 products, we will get the value of 75.00. Then we need to divide that value by ½ the number of samples. There are 30 samples, so we divide 75.00 by 15 and get 5.00. This tells us that the sample holds a 1Khz sine wave with an amplitude of 5.00 volts.



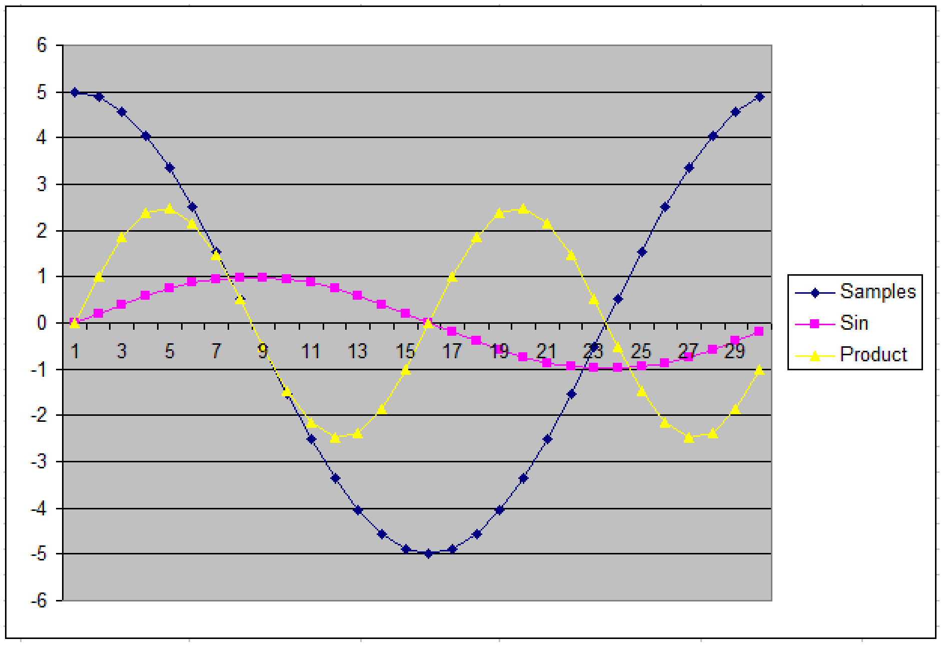
Let’s do the same process but attempt to detect any 2KHz sine wave within our sample.



Now you can see that the product [Yellow on graph] goes positive AND negative. And if you sum together all the points you will get zero. Indicating that there is no 2Khz sine wave component in the sample data.

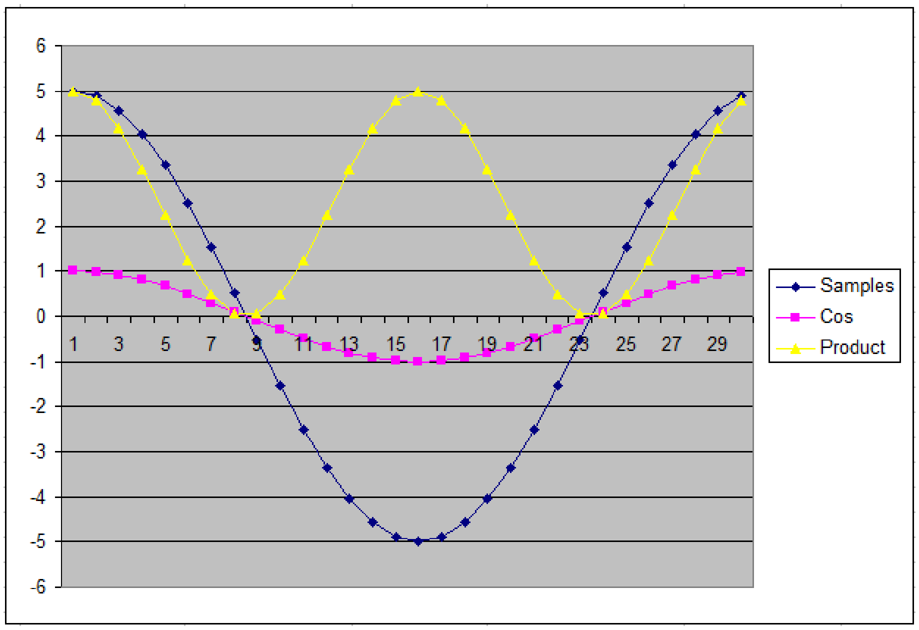
What if the sample data does not start at 0 degrees ?

Let say the sample data started at 45 degrees instead of at zero degrees.



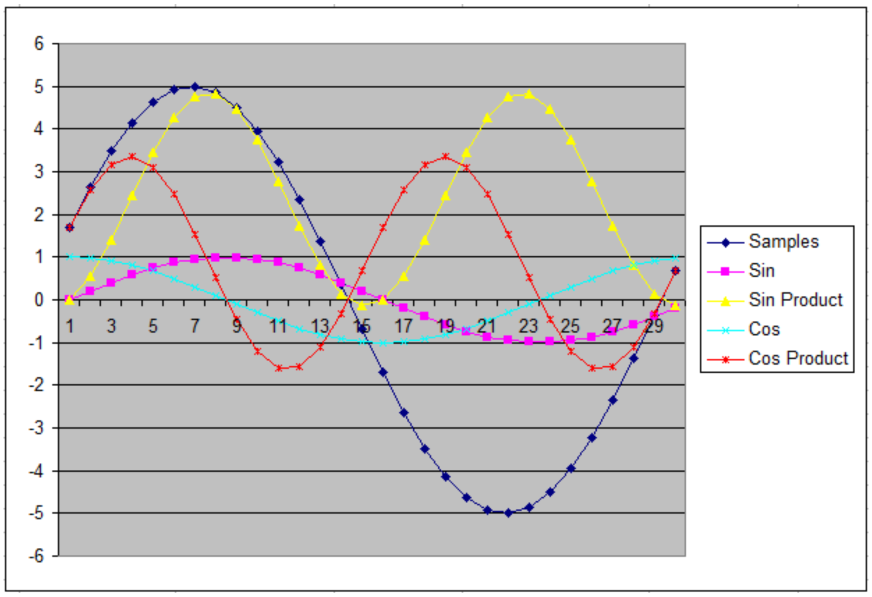
As you can see the product waveform goes both positive and negative. And in fact, if you sum the products you will get zero. Hmmm, THAT is a problem.

We solve the problem by doing the products twice, once with a sin wave, then again with a cosine wave. The product of the cosine wave (shown below) will look similar to our original product except that it is shifted.



Okay, so what about a sample that has some odd phase shift, say 20 degrees ?

It turns out that if you take the sum of the sin product, and the sum of the cos product and use them as X, Y points. The radius (hypotenuse) will be the amplitude and the angle (ATAN2) will be the phase offset.



In the data shown above the sum of the sin products = 4.69846 (after dividing by 15), and the sum of the cos products = 1.71010 (after dividing by 15). Finding the radius (hypotenuse) by using sqrt(4.69846^2 + 1.71010^2) = 5.000. And phase ATAN2(4.69846, 1.71010) = 20.0 degrees.

This tells us that our sample contains a 1KHz sin wave component with the amplitude of 5.0, and a phase of 20 degrees.

This example is of the very best case. There are all kinds of variables to overcome when doing DTF with real world data.

Performing a DTF “on the fly”.

One nice thing with the DTF is that it can be performed “on the fly”. That is, you don’t need a buffer to hold the sample data. As long the microcontroller is fast enough, the math can be performed on each sample as it is sampled in real-time.

Assume you have a Sin_Table and a Cos_Table

Assume you sample at 20910 hz looking for 697 Hz (1,2,3 in DTMF)

```
' Sin and Cos tables are Sin/Cos(angle) * 65536 / 15
Sin_Table := (0, 908, 1777, 2568, 3247, 3784, 4155, 4345,
              4345, 4155, 3784, 3247, 2568, 1777, 908,
              0, -908, -1777, -2568, -3247, -3784, -4155, -4345,
              -4345, -4155, -3784, -3247, -2568, -1777, -908)

Cos_Table := (4369, 4274, 3991, 3535, 2923, 2185, 1350, 457,
              -457, -1350, -2185, -2923, -3535, -3991, -4274, -4369,
              -4274, -3991, -3535, -2923, -2185, -1350, -457,
              457, 1350, 2185, 2923, 3535, 3991, 4274)

REPEAT
  SinSum := 0
  CosSum :=0
  REPEAT i FROM 0 TO 29
    Sample :=ReadSample ' -127 to +127
    SinSum := SinSum + Sample * Sin_Table[i]
    CosSum := CosSum + Sample * Cos_Table[i]

  SinSum := SinSum ~> 16
  CosSum := CosSum ~> 16
  SinSum := SinSum * SinSum
  CosSum :=CosSum * CosSum
  Ampl := Sqrt(SimSum + CosSum)
  PST.Dec(Ampl)
```