The Fourier Transform

It can be shown that signals can be described by a sum of sinusoidal components of various amplitudes, frequencies and phases. As we process these signals we need a way to determine the values of these components that make up the signal automatically and accurately.

The Fourier transform of a continuous function of time f(x) is a complex valued function of frequency, whose absolute value is the magnitude of frequency component present in the signal, while the complex argument represents the phase shift of the component. The transform is referred to as the frequency domain representation.

To show what we mean by a frequency domain representation, let us look at the example below. We have a sine wave with the frequency f. The graph on the left shows the time-domain representation of the signal. On the right is the frequency domain representation of the same signal. It has a single sinusoidal component with an amplitude of 2 and a frequency of f.

|  |  |
| --- | --- |
|  |  |

If we add more signals to the sinusoid in the time domain, we can see the results in the frequency domain below. We have added 2 additional sine waves to the initial wave giving us the final signal in green. By analysing the frequency domain, we can see the amplitude and frequency of each of the signal components within the Composite signal.

|  |  |
| --- | --- |
|  |  |

The frequency domain is showing us the frequencies where the energy of the signal is contained. The figure below is good visual representation of how the two are interlinked, which will be useful when we show what is happening mathematically.

For signal processing, we use the Discrete Fourier Transform (DFT) which deals with discrete functions of time:

Where , is the signal we are analysing, is the number of samples in the signal that is being analysed. is the sample number, which is the index of the time domain representation of the signal. Similarly, is the index into the sequence of frequency values that will be returned. This is referred to as the bin number.

Using Euler’s formula, we can rewrite the complex exponential in terms of sine and cosine functions, which will make it easier to visualise how this applies to signals. This results in the following:

The signal we are analysing is being multiplied by a set of cosine waveforms, and summed to give the real terms in the complex components in the results, while also being multiplied by a set of sine waveforms to give the imaginary terms. This process of multiplication and summation is known as correlation, [xxxxwhich we will look at in detail later. Suffice to say correlation][From the earlier section we racallxxx] is a measure of similarity between two signals, or a measure of the presence of one signal in another. This is essentially what the DFT is doing.

If we substitute into the formula above, we get the following:

When , , cos(0) = 1 and sin(0) = 0, so the formula becomes:

As the the average amplitude of is 0, then the value will be 0. The if the average amplitude deviates from 0, then the value of

When , becomes a cosine waveform with 1 cycle over samples and becomes a sine waveform with 1 cycle over samples. These signals are then multiplied with the original signal and the values are summed to give the associated real and imaginary term value of the frequency bin, . When , the signal is multiplied by waveforms with 2 cycles over samples to give the values for , and so on up to . This set of waveforms are called the analysis basis functions, and when we multiply these with the signal we are analysing, , we get a set of measurements from 0 to of the signals similarity with the cosine and sine waves, which indicates the presence of the cosine and sine waves within the signal .

|  |  |  |
| --- | --- | --- |
| k |  |  |
| 0 |  |  |
| 1 |  |  |
| 2 |  |  |
| … | ... | … |

These waveforms are called the analysis basis functions, and when we multiply these with the signal we are analysing, , we get a set of measurements from 0 to of the signals similarity with the cosine and sine waves, which indicates the presence of the cosine and sine waves within the signal .

As an example, we will use a 2Hz cosine signal , and set our sample rate, .

When we multiply this with the analysis basis functions above we get the following results:

|  |  |  |
| --- | --- | --- |
| k |  |  |
| 0 |  |  |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| … | ... | … |

Then, summing the values of each of the samples, we get the real and imaginary terms for the results of each of the frequency bins, which can be plotted in the frequency domain.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | … |
|  | 0+i0 | 0+i0 | **8+i0** | 0+i0 | … |

What is interesting about these results, is the only non-zero value lies in the bin where the basis function contains the same number of complete cycles as the signal being analysed. Another way of looking at is the value will be non-zero if the signal contains that base waveform.

We can apply the same process to signals comprised of multiple sinewaves and expect similar results. Using the signal and again set our sample rate, , we get the following results:

|  |  |
| --- | --- |
|  |  |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | … | |  | 0+i0 | 4+i0 | **16+i0** | 0+i0 | … | | |

Once again, we see the non-zero values at the components containing the base waveforms. The values of the magnitudes in the frequency domain appear to be much larger than in the time domain, however this is simply a result of the sampling frequency we have chosen. If you wish to scale the frequency domain amplitudes, simply divide by .

We will now introduce a phase shift into the signal being analysed to see how this affects the transform. Using the signal and again set our sample rate, . Once again, we run it through our analysis basis functions, but this time we start to notice values appearing in the imaginary terms.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | … |
|  | 0+i0 | 0+i0 | **6.23-i10.256** | 0+i0 | … |

Now, let’s try to verify these values. First, we need to scale the values by , giving us a value ~1.5575+i2.564. A cosine wave with a phase shift can be written in terms of a cosine wave plus a sine wave, . Knowing this, term can be rewritten as :

And this is what the DFT is telling us. The signal contains both a cosine component represented by the real value of the frequency bin value, and a sine component, represented by the imaginary term. The magnitude of this complex number, when scaled to account for the sample period, is equal to the amplitude of the original waveform, and the angle of this complex number in radians is the phase shift.

So far, we have looked at cases where the signals we are analysing have had an integer multiple number of cycles within the sampling period. However, things get a bit trickier when this is not the case.

Let’s look at a signal with a frequency of 2.4Hz, . When we run this through the transform with the sample period of 8 we get the results below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | … |
|  | 5.59 + i0 | 5.589 + i3.147 | 9.045 + i14.635 | 9.045 – i8.033 | … |
| Magnitude | 5.59 | 6.414 | 17.205 | 12.097 |  |

What we can see is the spectral energy is spread throughout the bins. This is known as spectral leakage. If we plot this in the frequency domain, we can see that a significant amount of the spectral energy is contained between bins 2 and 3.

The shape of this spectral leakage is well defined, and we will explore it later. Remember that the DFT is used to determine similarity between signals, but our signal does not exactly match the frequency of any of the analysis basis functions as the number of cycles is not an integer. However, there is a lot of similarity with the functions used for k = 2 and k = 3, as the frequency is between these two values. Because our signal doesn’t not have an integer number of complete cycles within the sample period, it is impossible to contain the spectral energy in a single bin, so we will always see spectral leakage in some form.

There are a number of techniques to make the results of spectral spreading more predictable and manageable. The first we will look at is known as zero-padding. As we have seen, signal being analysed is correlated against a set of analysis basis functions to produce a set of frequency bin values. The frequencies of these basis functions contain integer multiples of cycles that fit exactly into the sampling window. If the number of cycles in the of one of the sinusoidal components of the waveform matches exactly the number of cycles in one of these functions, the spectral energy with be contained in that one bin value. However, when the signal has a frequency that does not have an integer number of cycles in the sample period, there is no exact match and we get a spread of energy across the frequency bins, as we have seen above. Having more samples will increase the resolution of the frequency domain giving the DFT more opportunities of finding a signal which is an exact match for the signal frequency being analysed and pinpoint where the spectral energy lies. If we had an infinite number of samples, we could accurately detect all frequencies, however, this solution is not achievable. With zero-padding, the number of samples being analysed can be increased by appending a large number of zero valued samples to the end of the signal.

Let’s take the following example where we have a signal of 2Hz, with 1000 samples. We can see in the frequency domain there is a single spike, at bin number 2 and all other values are zero. We can also see that the magnitude of this bin is 250, which is the amplitude of the signal multiplied by half the number of samples. Now when we take the same signal, but increase the frequency to 2.5Hz, we see a spread of energy in the frequency domain, with most of the energy between bins 2 and 3, as should be expected, as the frequency of the signal lies between 2 and 3 cycles. We also notice that the magnitude of the maximum is much lower than it was then we matched the frequency exactly.

We now will zero-pad this signal with 1000 samples with 0 values and analyse the new signal. In the time domain, we can see our signal now has 2000 samples, with the original signal of 2.5 cycles up to sample 1000, and 1000 zero values. In our frequency domain plot, we now see a large, bell shape, known as the main lobe. Its maximum value is centred at bin 5, which corresponds to the 2.5Hz frequency of our signal. If the signal had multiple sinusoidal components, there would be a main lobe for each component. What we also see is the magnitude in back up to 250, as it was in the example with 2Hz over 1000 samples. However, we also notice a number of side lobes, which have been introduced as a result of the zero-padding. These can be problematic in accurately analysing the frequency spectrum data. We can reduce the size of these side lobes by using a process known as windowing.

In the example above, we zero padded by the length of the original signal. This is the minimum amount of zero-padding allowed. It worked in the example above as 2.5Hz over the period of 1000 samples did not have an integer value of cycles, but when we doubled the length of the sample period, there would be 5 full cycles of the waveform, so this matched the basis function associated with bin 5. If we use a value of 2.2Hz, zero-padding by 1000 samples would result in the broad spectral spread of energy we saw before. The solution in this case would be to zero-pad by a factor of 5 to allow the DFT find a waveform with the same number of cycles over the sample period as the signal. In practice, we do not know the values of the frequencies we are looking for, so the approach is to zero pad by as large a number as possible. This gives the DFT the greatest chance of finding the waveform that matches exactly with the frequency of the signal being analysed.

Windowing is applied to a signal to improve the behaviour in the frequency domain.  
it is achieved by multiplying a window function by the signal to be analysed. Finite-length sampling

Zero-padding is itself a form of windowing, where the window being applied is a rectangular window, which has a value of 1 for the first 0-N samples, and zero for values less than 0 and greater than N. For a moment, let us consider continuous signals. Our wave form is a continuous signal with the frequency 2.5Hz. This can be represented on a magnitude spectrum as a spike at the bin positions representing 2 Hz. When zero-padding, we multiply our wave by the rectangular window described above. This rectangular window has a frequency spectrum associated with it, with both positive and negative frequencies. When we multiply in the time domain, the effect in the frequency domain is a process called convolution. When we convolve the waveform associated with the window waveform, the result is this spectral shape associated with the rectangular window appearing everywhere there is a spike associated with the original frequency. This is introducing a large amount of high frequency components in the side lobes. If we had two sinusoids of similar frequencies in our signal close together, we can see the interference of spectral noise can lead to data that can be easily misinterpreted.

The ideal form of DFT would be an infinitely long sample period. In practical terms, we are restricted only by computing power, so we append a very large number of zeros. Going back to our 2.2Hz example, we will append 99000 zeroes, which for our purposes will act as the “infinite” resolution DFT response. Looking the frequency domain, we see that main lobe, with the peak at the bin associated with the 2.2Hz, and number of side lobes. If we plot the magnitude spectrum for the same signal padded with 9000 zeroes. We see that the samples align exactly with those from the “infinite” response. Doing this again with 4000, and 0 zeros appended, we see that all the values in the frequency bins are all aligned to the “infinite” curve. The shape of the response from the DFT with N samples will be a sampled subset of the infinite response.

What we try to achieve with windowing, is to use a function that has a spectral shape that is more manageable for analysis purposes. If we consider all There are many well defined window functions which can give us predictable responses in the frequency domain. Let’s take the Hanning window for an example.  
eoafuenaoufnaef BLEEDIN GRAPHS OOOOOOOOOO

The Hanning window can be described by the following equation:

It looks like this in the time domain. As we see, it has zero values at the edges and a peak in the centre. The magnitude spectrum of this window has a wide main lobe, and small side lobes. So what we expect to see when we apply this window to the signal, is this shape to appear at each frequency component in the original signal.

With windowing, we set the window to the same sample width as the signal we are analysing, and then multiply the signal

The resultant shape in the frequency domain is the bell-like shape we saw while zero-padding, but the side lobes have been greatly reduced. We also notice the main lobe is twice the width of the main lobe we saw with zero padding. We can now determine the amplitude, frequency and phase information of the sinusoid by analysing the magnitude spectrum. First, we need to find the bin with the maximum value. In this case, it is \*\*\*BINNUMBER-OIOINS\* and the magnitude is 125, which is half again of what it was when we were able to match a signal without the windowing. When using the Hanning Window, we divide the magnitude value of the maximum value by the length of the signal divided by 4. This depends on the window function used. The angle of the maximum gives us exactly the phase of the sinusoid. To get the frequency, multiply the bin number by the sampling frequency and divide by the number of samples.

We have seen that zero padding and windowing are useful in more accurately determining where spectral energy lies, but they come with overhead and a trade off must be found that maximises efficiency of processing, while also maximising accuracy of