FFT Spectrum Analysis (Fast Fourier Transform)
Additional information about FFT analysis

This training material will come around most topics of what is good to know in order to perform FFT analysis with high efficiency.

For additional information about FFT analysis please look at the links below:

- FFT Analysis (Fast Fourier Transform): The Ultimate Guide to Frequency Analysis
- FFT analyzer Online (F1) help information
- Processing Markers Online (F1) help information
- FFT analyzer Application page
What is frequency analysis?

Frequency analysis is just another way of looking at the same data. Instead of observing the data in the time domain, with some not very difficult, yet inventive mathematics frequency analysis decomposes time data in the series of sinus waves.

We can also say that frequency analysis checks the presence of certain fixed frequencies.

The image below shows the signal, which consists of three sine waves with the frequencies of 0.5 Hz, 1 Hz, and 2 Hz, and then on the right side the decomposed signal.
Just to make those sine waves better visible, let us show them in a nicer way. On the x-axis, there are frequencies and on the y-axis, there are amplitudes of the sine waves.

And this is really what the frequency analysis is all about: showing the signal as the sum of sinus signals. And the understanding, how that works, helps us to overcome problems that it brings with it.
Fourier transform

The mathematician Fourier proved that any continuous function could be produced as an infinite sum of sine and cosine waves. His result has far-reaching implications for the reproduction and synthesis of sound. A pure sine wave can be converted into sound by a loudspeaker and will be perceived to be a steady, pure tone of a single pitch. The sounds from orchestral instruments usually consist of a fundamental and a complement of harmonics, which can be considered to be a superposition of sine waves of a fundamental frequency f and integer multiples of that frequency.

Fourier analysis of a periodic function refers to the extraction of the series of sines and cosines which when superimposed will reproduce the function. This analysis can be expressed as a Fourier series.

Fourier series

Any periodic waveform can be decomposed into a series of sine and cosine waves:

\[ f(t) = a_0 + \sum_{n=1}^{\infty} a_n \cdot \cos \left( \frac{2\pi nt}{T} \right) + \sum_{n=1}^{\infty} b_n \cdot \sin \left( \frac{2\pi nt}{T} \right) \]

where \( a_0, a_n, \) and \( b_n \) are Fourier coefficients:

\[ a_0 = \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \]

\[ a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos \left( \frac{2\pi nt}{T} \right) dt \]

\[ b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin \left( \frac{2\pi nt}{T} \right) dt \]

Discrete Fourier transform

For discrete data, the computational basis of spectral analysis is the discrete Fourier transform (DFT). The DFT transforms time-based or space-based data into frequency-based data.

The DFT of a vector \( x \) of length \( n \) is another vector \( y \) of length \( n \):
where \( w \) is a complex \( n \)th root of unity:

\[
y_{p+1} = \sum_{j=0}^{n-1} \omega^{jp} x_{j+1}
\]

We used \( i \) for the imaginary unit and \( p \) and \( j \) for indices that run from 0 to \( n-1 \). The indices \( p+1 \) and \( j+1 \) run from 1 to \( n \).

Data in the vector \( x \) are assumed to be separated by a constant interval in time or space, \( dt = 1/fs \) or \( ds = 1/fs \), where \( fs \) is the sampling frequency. The DFT \( y \) is complex-valued. The absolute value of \( y \) at index \( p+1 \) measures the amount of the frequency \( (f = p(fs / n)) \) present in the data.

The first element of \( y \), corresponding to zero frequency, is the sum of the data in \( x \). This DC component is often removed from \( y \) so that it does not obscure the positive frequency content of the data.

An example of this is the square wave in the picture below. A square wave is composed of an infinite summation of sinusoidal waves.

![Square wave displayed in time (above) and in the frequency domain (below)](image)

Let’s think about how the equation for discrete Fourier transform works:
To check the presence of a certain sine wave in a data sample, the equation does the following:

\[
X(k \omega_0) = \sum_{n}^{N-1} x[n] \cdot (\cos(2\pi kn/N) + i \cdot \sin(2\pi kn/N))
\]

To check the presence of a certain sine wave in a data sample, the equation does the following:

1. Multiplies the signal with a sine wave of that frequency which we want to extract. The image below shows the signal (black line), which consists only of a sine wave with 50 Hz. We try to extract the 36 Hz on the left side and 50 Hz on the right side (they are shown as blue lines). Light blue filled wave shows multiplied values.

2. Multiplied values are summed together and this is the main trick. If there is a component in a signal like in the right picture the multiplication of positive signal parts and extraction sine waves gives the positive result. Also, the multiplication of negative signal parts and negative extraction sine waves gives positive results (observe the right image). In this case, the sum of the multiplied sine waves will be nonzero and will show the amplitude of the 50 Hz part of the signal. In the case of 36 Hz, there are both positive and negative sides of multiplication values and the sum will be (almost, as we will see further on) zero.

3. And that’s it. That sum gives the estimate of the presence of frequencies in the signal. We check sine and cosine to get also phase shift (in the worst case if the phase shift would be 90 degrees, the sum of sine functions would always give zero).

The principle shown above can extract basically any frequency from the sine wave, but it has one disadvantage - it is awfully slow. The next important step in the usage of DFT was the FFT algorithm - this analysis reduces the number of calculations by rearranging the data. The disadvantage is only that the data samples must be of length, which is the power of two (like 256, 512, 1024 and so on). Apart from that, the result is practically the same as for the DFT.
FFT - Fast Fourier Transform

Fast Fourier transform is a mathematical method for transforming a function of time into a function of frequency. It is described as transforming from the time domain to the frequency domain.

The Fast Fourier transform (FFT) is a development of the Discrete Fourier transform (DFT) which removes duplicated terms in the mathematical algorithm to reduce the number of mathematical operations performed. In this way, it is possible to use large numbers of samples without compromising the speed of the transformation. The FFT reduces computation by a factor of $N/(\log_2(N))$.

FFT computes the DFT and produces exactly the same result as evaluating the DFT; the most important difference is that an FFT is much faster!

Let $x_0, ..., x_{N-1}$ be complex numbers. We have already seen that DFT is defined by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \ldots, N - 1.$$  

Evaluating this definition directly requires $N^2$ operations: there are $N$ outputs of $X_k$, and each output requires a sum of $N$ terms. An FFT is any method to compute the same results in $N \log(N)$ operations. All known FFT algorithms require $N \log(N)$ operations.

To illustrate the savings of an FFT, consider the count of complex multiplications and additions. Evaluating the DFT’s sums directly involves $N^2$ complex multiplications and $N(N-1)$ complex additions. FFT algorithm can compute the same result with only $(N/2)\log_2(N)$ complex multiplications and $N\log_2(N)$ complex additions.

<table>
<thead>
<tr>
<th></th>
<th>DFT</th>
<th>FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex multiplications</td>
<td>$N^2$</td>
<td>$(N/2)\log_2(N)$</td>
</tr>
<tr>
<td>complex additions</td>
<td>$N(N-1)$</td>
<td>$N\log_2(N)$</td>
</tr>
</tbody>
</table>

In practice, actual performance on modern computers is usually dominated by factors other than the speed of arithmetic operations and the analysis is a complicated subject, but the overall improvement from $N^2$ to $N \log_2(N)$ remains.

On the image below, you can see original data of a signal in the time domain (units in seconds [s]), and data after Fast Fourier transformation in the frequency domain (units in hertz [Hz]).
Once you know the harmonic content of a signal from Fourier analysis, you have the capability of synthesizing that signal from a series of pure tone generators by properly adjusting their amplitudes and phases and adding them together. This is called Fourier synthesis.
Autospectra and Cross-spectra

Autospectrum

An autospectrum or auto power spectrum is a function commonly explored both in signal and system analysis. It is computed from the instantaneous (Fourier) spectrum as:

\[ A(f) = |A(f)| \cdot e^{j \phi_A(f)} \]

\[ A^*(f) = |A(f)| \cdot e^{-j \phi_A(f)} \]

\[ S_{AA}(f) = E[A(f) \cdot A^*(f) \cdot e^{j \theta}] = E[|A(f)|^2] \]

It is computed from the instantaneous spectra of both channels. All other functions are computed during post-processing from the cross-spectrum and the two auto spectrums - all functions are the functions of frequency.

Cross-spectrum

A cross spectrum or cross power spectrum is based on complex instantaneous spectrum \( A(f) \) and \( B(f) \), the cross-spectrum \( S_{AB}(f) \) (from A to B) is defined as:

\[ A(f) = |A(f)| \cdot e^{j \phi_A(f)} \]

\[ B(f) = |B(f)| \cdot e^{j \phi_B(f)} \]

\[ S_{AB}(f) = E[A(f) \cdot B(f) \cdot e^{j(\phi_B(f) - \phi_A(f))}] \]
The amplitude of the cross-spectrum $S_{AB}$ is the product of amplitudes, its phase is the difference between both phases (from A to B). Cross spectrum $S_{BA}$ (from B to A) has the same amplitude, but opposite phase. The phase of the cross-spectrum is the phase of the system as well.

The cross-spectrum itself has little importance, but it is used to compute other functions. Its amplitude $|G_{AB}|$ indicates the extent to which the two signals correlate as the function of frequency and phase angle of $G_{AB}$ indicates the phase shift between the two signals as the function of frequency. An advantage of the cross-spectrum is that influence of noise can be reduced by averaging. That is because the phase angle of the noise spectrum takes random values so that the sum of those several random spectra tends to zero. It can be seen that the measured autospectrum is a sum of the true autospectrum and autospectrum of noise, whilst the measured cross-spectrum is equal to only the true cross-spectrum:

\[
\]

channel A autospectrum:

\[
\]

channel B autospectrum:

\[
\]

cross spectrum:

One- and two-sided spectra

Both auto spectra and cross-spectrum can be defined either as two-sided (notation $SAA$, $SBB$, $SAB$, $SBA$) or as one-sided (notation $GAA$, $GBB$, $GAB$, $GBA$). One-sided spectrum is obtained from the two-sided one as:
\[ G_{AB}(f) = \begin{cases} 
0 & \text{for } f < 0 \\
S_{AB}(f) & \text{for } f = 0 \\
2S_{AB}(f) & \text{for } f > 0 
\end{cases} \]
Properties of Fourier transform

In the image below, we can see a typical FFT screen. The maximum frequency of the FFT is half of the signal sampling frequency (in this case the sample rate was 22000 samples/sec), but in the upper region the results are never reliable, so the sampling result should be set to:

\[ SampleRate = \text{MaximumSignalFrequency} \cdot 2 \cdot 1.25 \]

1.25 is the absolute minimum factor for also getting the right values in the upper region of the FFT. A factor of 1.28 is commonly used in signal analysis in order to obtain a 'nice' Analysis Bandwidth (also referred to as Frequency Span). For example having a sample rate given by:

\[ SampleRate = 2^{15} \text{Hz} = 32768 \text{Hz}, \]

then:

\[ f_{\text{Span}} = \frac{\text{Samplerate}}{2 \cdot 1.28} = \frac{32768 \text{Hz}}{2.56} = 12800 \text{Hz} \]

The factor of 2 comes from the famous Nyquist criteria (or more correctly from the Nyquist–Shannon sampling theorem), which says that maximal signal frequency adequately presented in the digitized wave is the half of the sampling rate.

The result of FFT is a set of amplitudes of certain frequencies. The amount of amplitudes in the set is given by the Number of Lines parameter for the FFT. The Number of Lines parameter is user-selectable, and it determines the resolution of the FFT.
Line resolution is a change in frequency between two frequency lines, which are extracted from the signal and is calculated with the equation:

\[
\text{Line Resolution} = \frac{\text{Sample Rate} / 2}{\text{Number of Lines}}
\]

So the question is: why not always use the maximum number of available frequency lines, which gives more exact results? The answer is simple: because, with more frequency lines it takes more time to calculate FFT spectra.

\[
\text{Time To Calculate} = \frac{\text{Number of Lines} \cdot 2}{\text{Sample Rate}}
\]

Just for fun we can also combine the equations above and we get:

\[
\text{Line Resolution} = \frac{1}{\text{Time To Calculate}}
\]

Let's look at the equations above and make a list for the 22 kHz sample rate:

<table>
<thead>
<tr>
<th>Number of lines</th>
<th>Line resolution [Hz]</th>
<th>Calculation time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>21,5</td>
<td>0,046</td>
</tr>
<tr>
<td>1024</td>
<td>10,75</td>
<td>0,093</td>
</tr>
<tr>
<td>4096</td>
<td>2,685</td>
<td>0,372</td>
</tr>
<tr>
<td>16384</td>
<td>0,67</td>
<td>1,49</td>
</tr>
</tbody>
</table>

So the number of lines combined with the sample rate also defines the speed of the FFT when non-stationary signals are applied. With more lines, FFT will appear slower and changes in signal will not be shown that rapidly.

Different amplitude scales of FFT can reveal more about the signal if used correctly. Linear amplitude scale gives the best view of maximum peaks in the signal, a logarithmic amplitude scale can show more invisible peaks and signal noise but gives a worse comparison of high and low peaks. Scale in dB gives the best estimation of signal noise if 0 dB is maximum measurable value and is also used in noise measurements, where the dB scaling is actually the result since the human ear has logarithmic sensitivity to noise.
X scale can be either linear or logarithmic. Linear scaling is the correct representation of the mathematic transformation and usually gives the best information for analysis. Sometimes like in the example shown in the picture above it is nice to see the x-axis in logarithmic values since most interesting frequencies are in a lower region. We have to know that just to set the x scale to logarithmic does not enhance the results in the lower region, so the resolution will be better in the upper region since there are more frequency lines available there.
If we use another technique, called CPB (constant percentage bandwidth), also referred to as Octave Analysis, this will give us the same resolution in all regions when the x-axis is logarithmic. This is achieved by the fact that upper region lines cover wider frequency ranges than the lower one.

The resolution of the bands is defined by $1/n$ description, where $n$ is the number of bands in one octave. The most widely used is the 1/3 octave analysis, which is the standard for noise measurements. 1/12 and even better 1/24 octave analysis already gives good resolution also for signal analysis.
Windowing functions

If a sine wave is not located directly on an FFT frequency line, we get amplitude values on both sides of the main band. Such amplitudes can be pretty high and affect FFT results, (with no window function, it can be about 10% of the original values for about 10 neighbor lines). If there is another sine wave in the signal in this region, which is lower than this 10%, it will be completely hidden by the leakage effect.

This is a phenomenon that occurs because the FFT algorithm can only be applied to periodic signals so the sampled input signal is 'periodized'. If the sampled signal is not periodic, or an integer number of periods is not sampled, discontinuities occur in the periodic signal processed by the FFT, causing the energy contained in the signal to 'leak' from the signal frequency bin into adjacent frequency bins. This leakage causes amplitude errors in the frequency spectrum.

As a result of the amplitude errors caused by spectral leakage, small frequency peaks will occur close to larger ones.

Window functions are used to reduce the effects of spectral leakage. Windowing is used to assign a weighting coefficient to each of the input samples, reducing those samples that cause spectral leakage. In effect, samples at the beginning and at the end of the sampling period are reduced to zero so that the discontinuities in the periodized sampled signal are removed.

In the picture below we can see the effect of windowing in a signal.
On the picture below we can see a spectrum of a signal without spectral leakage, spectrum with spectral leakage, and spectrum with windowing.

Windows are characterized by a number of properties as shown in the picture below.
The shape of the window’s main lobe is defined by the -3 dB and -6 dB main lobe width. These are defined as the width of the main lobe, in frequency bins, where the window response becomes respectively 3 dB or 6 dB less than the main lobe peak gain. The width of the main lobe of the frequency spectrum is important, as it affects the frequency resolution of the window (ability to distinguish between closely spaced frequency components). As the main lobe narrows, frequency resolution increases. However, with this narrowing of the main lobe, the window energy spreads into the side lobes, increasing the spectral leakage. Therefore, a compromise between frequency resolution and spectral leakage must be reached.

The maximum sidelobe level is defined as the level, in decibels, of the maximum side lobe, relative to the main lobe peak gain.

Sidelobe roll-off rate is the rate of decay of frequency of the side-lobe peaks, in decibels per decade.

The choice of the window depends upon the frequency content of the signal. A popular choice is the Hanning window. This window has quite a narrow main lobe, therefore, good frequency resolution and reasonable side lobe suppression making it suitable for many applications. Blackman-Harris window has excellent sideband rejection with an acceptably narrow main lobe.
Fourier transformation errors

The theoretical discrete Fourier transformation (DFT) has absolutely no error. The only problem is that the sum goes from minus infinity to plus infinity. Because we live in a fast-paced world we don't have the time to wait that long so we run into problems.

Amplitude error (picket-fence effect)

FFT results based on FFT time blocks can produce 'non-null' results even when the signal does not correspond to the frequencies extracted from the signal, since the pure frequencies are 'leaked' over to neighbor frequencies - a consequence of the finite FFT time block duration $T$ used. For the same reason, if the frequency does not fall exactly on the frequency line, the amplitudes seem to be lower. This is called the 'picket fence' effect.

Let's look at the picture below - 10 Hz and 12 Hz are the exact frequency lines. In the example, there are 10 Hz and 12 Hz sine waves marked as black, which are transformed correctly, and there are also frequencies in between which have lower amplitudes. Maximum amplitude error can go up to 35% of the correct value.

For amplitude errors, a bunch of people tried to minimize that problem. Those were Hamming, Hanning, Blackman, Harris, and others. They have created an assortment of functions, which try to correct the errors. Window functions are multiplied with the FFT blocks of the original time-domain signal and since they are usually 0 at the beginning and the end, sine waves could also
be in-between lines or phase-shifted and they will leak less over neighbor frequencies - less discontinuity at the FFT block ends.

The picture below shows some of these functions in the time domain.

And here is the most common question to FFT: what are the differences between windows and when to use certain windows?

The rule of thumb is that when we want a pure transformation with no window’s side effects (for advanced calculations), we should use a Rectangular window (which is equal to no window).

For general-purpose, Hanning or Hamming are commonly used because they provide a good compromise between fall-off and amplitude error (maximum of 15%). This comes from the fact that old frequency analyzers didn’t have that many possibilities in terms of frequency lines and these two windows have a narrow sideband.

When a more dynamic range is necessary (we want to see very small signals among large ones), Blackman or Kaiser’s window is a better choice because sidebands are 10 times lower than with the Hanning window. However, the sideband width is wider. Here it comes to the point - if more lines in FFT are chosen, we can use these windows and still larger sidebands had no real disadvantage.

If correct amplitudes are needed, we should use the flat-top window. The amplitudes would be wrong by only a fraction (as low as 1%). Of course, there is a penalty - neighbor frequencies are also very high (sideband width is high). This window is most...
suitable for calibration. But here it is the same: if a great number of lines are used, then this is often a minor problem. Just remember that the FFT block length \( T \) is the reciprocal of the line spacing, so with more spectral lines it requires a longer time duration per spectrum being calculated.

![Window characteristics](image)

Window characteristics (maximum amplitude error, sideband width, highest sideband attenuation, and sideband slope attenuation) are best described in the picture below. We have already discussed the maximum amplitude error: it is an error of amplitude if the sine waves do not fall on the frequency line. Windows try to eliminate this problem and because of that, they widen the first band. The sine waves are no longer on one line in FFT but spread along several lines. The ability to recognize small sine waves among larger ones is determined by the highest sideband attenuation and the sideband slope attenuation. These two values determine the leakage of the FFT and that’s nicely seen in the picture below. For example, if there is a signal with a frequency of 30 Hz and an amplitude of 0.0001, we would never see it because the 10.5 Hz signal has bigger leakage than the requested frequency signal. But if the rectangular window is used, we would never even see the signal with an amplitude of 0.01.
For different kinds of windows, the table below shows the values of all window properties. This is a numerical representation of the above-mentioned rules.

<table>
<thead>
<tr>
<th>Window type</th>
<th>Maximum amplitude error [%]</th>
<th>Width of the first band [line]</th>
<th>Highest sideband [%]</th>
<th>Sideband slope [dB/decade]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangular</td>
<td>36</td>
<td>2</td>
<td>22</td>
<td>-20</td>
</tr>
<tr>
<td>Hanning</td>
<td>15</td>
<td>2</td>
<td>2,5</td>
<td>-60</td>
</tr>
<tr>
<td>Hamming</td>
<td>18</td>
<td>2</td>
<td>0,7</td>
<td>-20</td>
</tr>
<tr>
<td>Blackman</td>
<td>12</td>
<td>3</td>
<td>0,12</td>
<td>-60</td>
</tr>
<tr>
<td>Flat top</td>
<td>0,02</td>
<td>5</td>
<td>0,0023</td>
<td>-20</td>
</tr>
</tbody>
</table>

The image below shows zoomed FFT of a pure sine wave, which fits the frequency line exactly. Abscissa axis shows the value of the line. In normal FFT, only values of the 0, 1, 2, etc. are calculated, so only those values are shown in the FFT. We can see the width of the first sideband, the highest sideband, and the sideband attenuation very clearly.
If a sine wave signal frequency falls between two lines, we see only the values on each side of the real frequency with sidebands energies that are dependent on the relative frequency deviation between the FFT line and the real frequency. This is best seen if we take a function generator, set the frequency to an exact frequency line, set the amplitude scaling to logarithmic and the FFT will look fantastic. No leakage, exact amplitude. Now switch the frequency from the function generator to the one between two lines in the FFT and the result will be just terrible: large amplitude errors, huge leakage.

There is one more trick with windows: if we are sure that all the frequencies will fall on their frequency lines, a rectangular window will give us the best result. For example to measure the harmonics of the power line (having a fundamental frequency at 50 Hz in Europe), choose 6400 or 9600 sample/sec sampling rate, so that the line resolution will give exact 50, 100, 150 Hz... FFT lines, then choose a rectangular window and observe the perfect result in the Y log scale.
Aliasing

Another problem arises due to the signal conditioning. As mentioned earlier, the sample rate has to be at least twice the maximum signal frequency due to Nyquist–Shannon sampling theorem, else aliasing effects will occur. The image below shows the reason for it. Vertical lines represent samples taken with A/D converter and the black line is the original signal. But if we look at the orange line, which is the signal from the A/D converter, the signal is totally wrong because too few samples per period were taken to correctly represent the signal.

Of course, the problem above is not an FFT problem, but it is very important to know how to correctly identify the cause of the error. And sometimes there are some lines in FFT, which can be only explained in terms of aliasing. In FFT, if we change the frequency to the ranges above the maximum frequency limit, that line will not disappear but will bounce back and will show a fake frequency.

Aliasing example

To see that effect, a function generator and Dewesoft SIRIUS HS (High Speed) set to use no anti-aliasing filter are used, and the FFT analyzer perfectly shows the problem.

The signal is sampled with 1 kHz.

On the upper left side of the screen, we can see the FFT of the signal recognized by hardware with no anti-aliasing filter. On the
The first output frequency from a function generator was 400 Hz. Also, the frequency detected by our hardware was 400 Hz.

The second output frequency was 500 Hz (exactly half of our sampling rate). We can see that the hardware with no anti-aliasing filter detects a frequency at 0 Hz (DC). This is because of the described Nyquist theorem.
The third output frequency was 600 Hz. We can clearly see that the signal above 500 Hz bounces back. Our hardware detected a signal with a frequency of 400 Hz.

For the problem of aliasing, there is not much to be done in the FFT domain. Actually, there is absolutely nothing we can do when the samples have already been taken. So the first thing to do would be to choose the A/D board which has anti-aliasing
filters in the front, the second thing to do would be to use external filters or we can simply set the sampling rate to more than twice the maximum frequency present in the signal.
Averaging of the signal

To enhance the result, we can use averaging of the signal in the frequency domain. Averaging means that we calculate multiple FFT spectra and average their individual frequency lines.

There are many ways to average the signal, but the most important are Energy (RMS) averaging with Linear or Exponential weighting:

- **Energy** - is Linear RMS averaging, where each FFT spectrum counts the same in the results, and the result is the square root of the power mean of spectral values.
- **Energy (Exp.)** - is Exponential RMS averaging - where the FFT spectra are weighted less and less over time, and the result is the square root of the power mean of spectral values.

Next to these averaging types Dewesoft also support Maximum hold of spectral values. This Maximum hold type does actually not average spectral values, but instead it keeps individual peak values across multiple spectra. The result is a spectrum of peak line values coming from a mix of spectra.

Dewesoft also support an averaging type called Linear, which perform linear but not RMS averaging. This averaging type is rarely used and for most applications instead the Energy (linear RMS) averaging type would be the correct choice.

There is one more thing about the averaging: loss of information. When averaging is used with window functions, we could lose some data due to the window multiplication effects. In the image below, there is one example where the signal only consists of one pulse. If we average the result, use the window function and we are unlucky, the signal will fall in the region where the window sets the values to zero, and in the resulting FFT, we will never see this pulse.

![Graphical representation of overlap function](image)
That's why there is a procedure called overlapping which overcomes this problem. It no longer calculates averages one after another but takes some part of the time signal, which is already calculated and uses it again for calculation. There could be any number for overlap, but usually, there is 25%, 50%, 66.7%, and 75% overlapping.

50% overlapping means that the calculation will take half of the old data. Now all data will be for sure shown in the resulting FFT.

With 66.7% and higher overlapping, every sample in the time domain will count exactly the same in the frequency domain, so if it's possible, we should use this value for overlapping to get mathematically correct results.

Real-time frequency analyzer

What does a 'real-time' frequency analyzer mean? It means that it is able to calculate and show data with 66.7% overlapping and, therefore, has no data loss.
Representation of different signals in the FFT

All signals that are periodic in time but are not pure sine waves, produce base harmonic components as well as additional higher harmonics. The more the signal is not like a pure sinusoid, the greater higher harmonic components become.

A harmonic of a wave is a component frequency of the signal that is an integer multiple of the fundamental frequency \( f \), the harmonics have frequencies \( 2f, 3f, 4f, \ldots \). The harmonics have the property that they are all periodic at the fundamental frequency. If the fundamental frequency (first harmonic) is 25 Hz, the frequencies of the next harmonics are 50 Hz (second harmonic), 75 Hz (third harmonic), 100 Hz (fourth harmonic), etc.

1.) Triangle, rectangular

On the left side, in the picture below we can see a Triangle signal in the time domain and on the right side is the Triangle signal in the frequency domain.

Triangle signal in time and in the frequency domain

On the left side, in the picture below we can see a Rectangular signal in the time domain, and on the right side is the Rectangular signal in the frequency domain.

Rectangular (square) signal in time and in the frequency domain
2.) Impulse

An impulse is quite an interesting thing - it cannot be described as a sum of sine waves. Or in other words: it is shown equally on all of the frequency lines. That’s the reason why we use it as the basic excitation principle to get frequency responses of the system. Other common excitation types are swept sine and different types of noise, but this is already a part of another story - dual-channel frequency analysis and modal testing.

On the left side, in the picture below we can see the Impulse signal in the time domain, and on the right side is the Impulse signal in the frequency domain.

![Impulse signal in time and in the frequency domain](image)

3.) White noise

The theory says that white noise consists of all frequencies. That’s why the infinite frequency spectrum of the white noise is the straight line. The shorter the samples are, the more different amplitudes for certain frequencies we get in the noise level. To get a fixed noise line averaging must be used. The picture below shows an already averaged FFT of the white noise.

On the left side, in the picture below we can see White noise in the time domain, and on the right side is White noise in the frequency domain.

![White noise signal in time and in the frequency domain](image)
4.) Beating (two closely spaced signals)

Beating in the time domain is somehow hidden and looks like one frequency with changing amplitudes. Only FFT will reveal two frequency lines if a high enough line resolution is chosen. The difference between the two frequencies is the modulation frequency shown in the time domain.

On the left side in the picture below we can see a beating signal in a time domain, and on the right side is the beating signal in the frequency domain.

![Beating signal in time and in the frequency domain](image)

5.) Amplitude modulated signal

The amplitude modulated (AM) signal is shown as two sideband frequencies. The difference between the base frequency and the sideband frequency is the modulated frequency (10 Hz, in this case) also seen clearly in the time domain. The rule here is the same as with beating - to reveal the modulation; we should choose high enough line resolution. In fact, the time signal or FFT time block, which is the base for the FFT calculation, should show some modulation peaks in order to achieve a proper spectral resolution. When windowing is used the window function will smear the frequency line resolution, such that amplitudes of the sideband frequencies will overlap with the amplitude of the main band frequency if not the FFT time block duration T covers enough modulation peaks - to achieve a sufficient amount of spectral lines between the sidebands and the main band.

On the left side, in the picture below we can see Amplitude modulated signal in the time domain, and on the right side is Amplitude modulated signal in the frequency domain.
Amplitude modulated signal in time and in the frequency domain
FFT analysis module in Dewesoft
Example of measurement with FFT analyser

For a measurement example, we used a blue toy with an electromotor and an encoder. An accelerometer was placed on the housing of the toy. When we run the machine up to 3000 RPMs the machine vibrates.

To observe the behavior of the machine we add an FFT analyzer. The input signal is an accelerometer signal that is attached to the rotating machine.
Before we run the machine, let’s add a visual control, the 3D graph. Select the design button and add a 3D graph widget.

![Adding a new 3D graph widget](image)

The next step is to select the channel that will be shown in the graph. In our example, it was the signal from the FFT analyzer.

The graph shows the amplitude [m/s²] plotted against frequency [Hz]. When we run the machine, we clearly see the first harmonic (values indicated with a red color):
By using the 3D graph widget instead of a 2D graph we can see how the harmonics are evolving over a time axis.

With the 3D graph different types of projections can be selected under the widget properties, as shown below:

Projection settings for the 3D graph display widgets
FFT markers