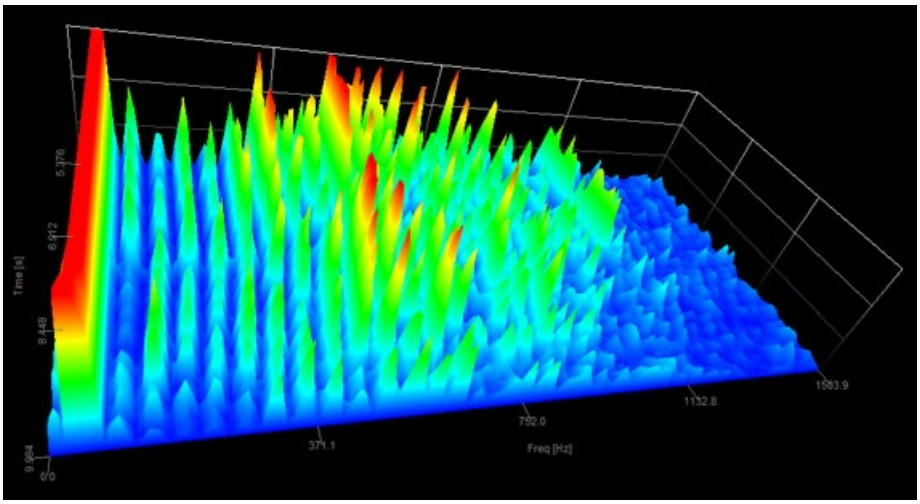
FFT Spectrum Analysis (Fast Fourier Transform)



What is frequency analysis?

For cyclical processes, such as rotation, oscillations, or waves, frequency is defined as a number of cycles per unit of time. For counts per unit of time, the SI unit for frequency is hertz (Hz); 1 Hz means that an event repeats once per second.

The time period (T) is the duration of one cycle and is the reciprocal of the frequency (f):

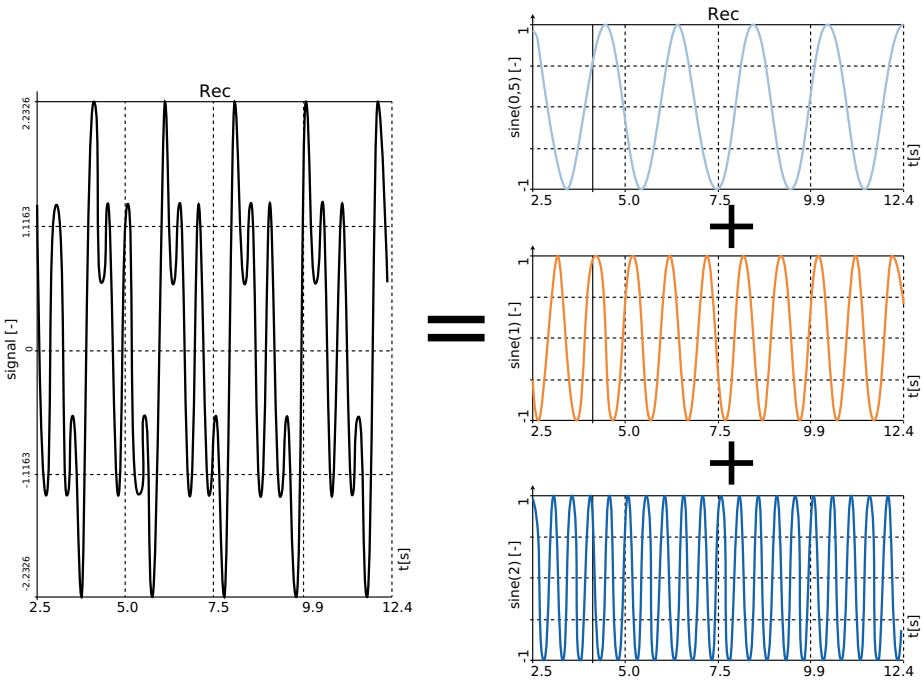
$$T = \frac{1}{f}$$

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.



DEWESoft®
measurement innovation

Image 1: Signal consisting of three sine waves with different frequencies

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.

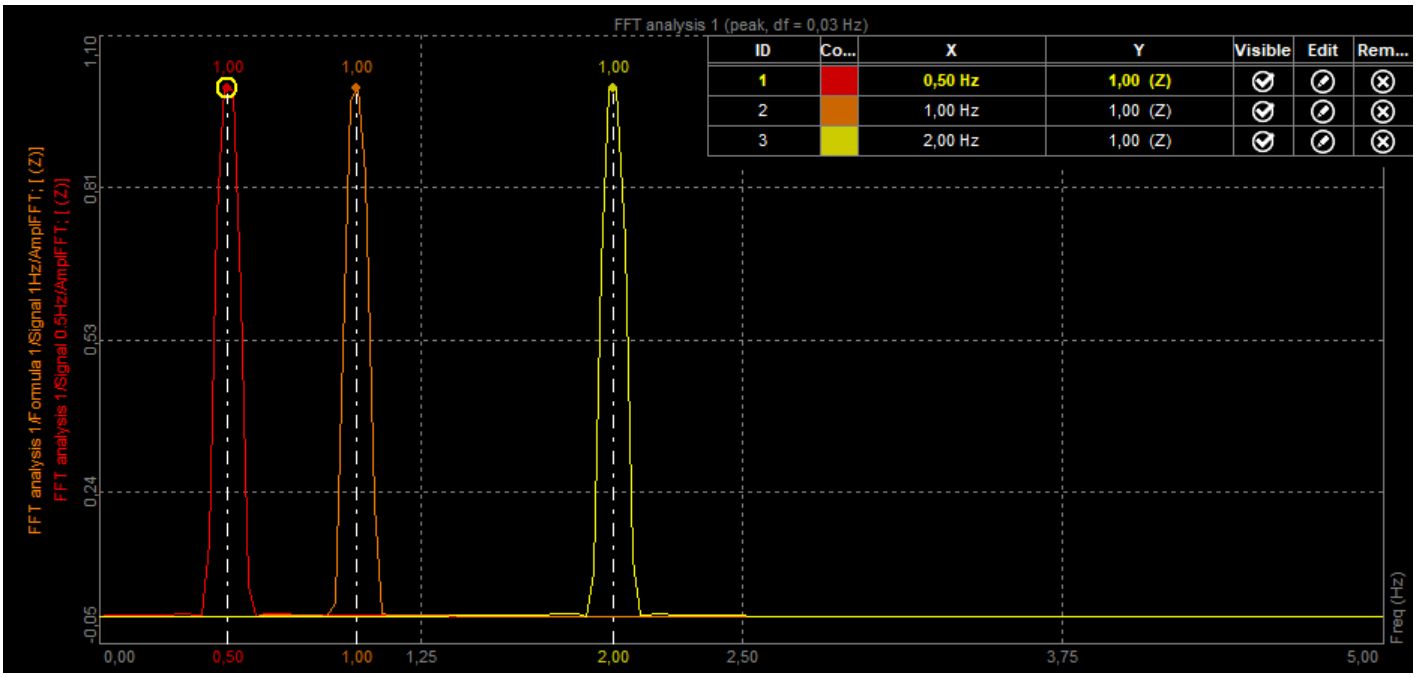


Image 2: Frequency representation of sine waves with different frequencies

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:

$$f(t) = a_0 + \sum_{n=0}^{\infty} a_n \cdot \cos\left(\frac{2\pi nt}{T}\right) + \sum_{n=0}^{\infty} b_n \cdot \sin\left(\frac{2\pi nt}{T}\right)$$

where a0, an, and bn are Fourier coefficients:

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

,

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

,

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

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:

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

where w is a complex nth root of unity:

$$\omega = e^{-2\pi i/n}$$

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.

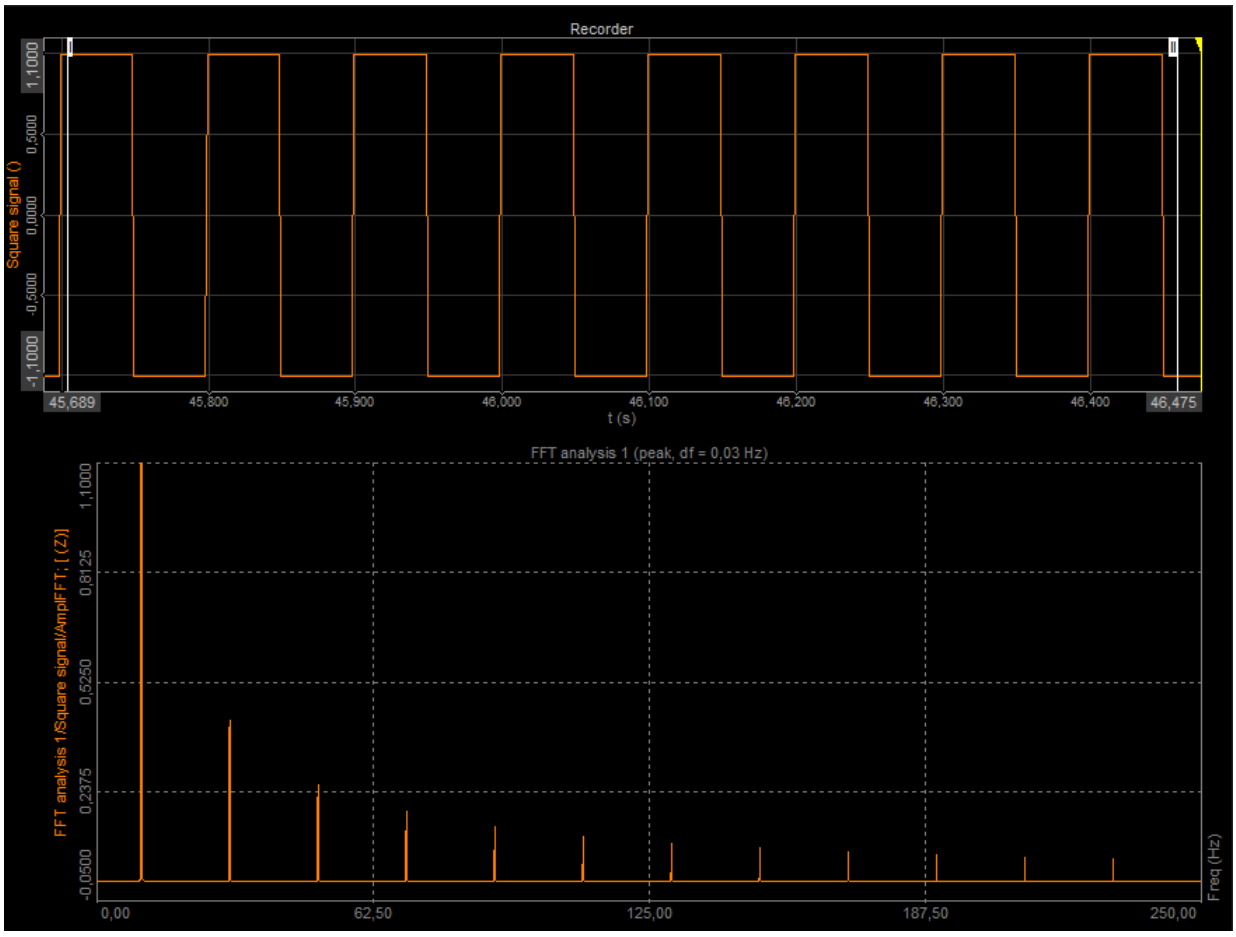


Image 3: Square wave displayed in time (above) and in the frequency domain (below)

Let's think about how the equation for discrete Fourier transform works:

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

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 deg, the sum of sine functions would always give zero).

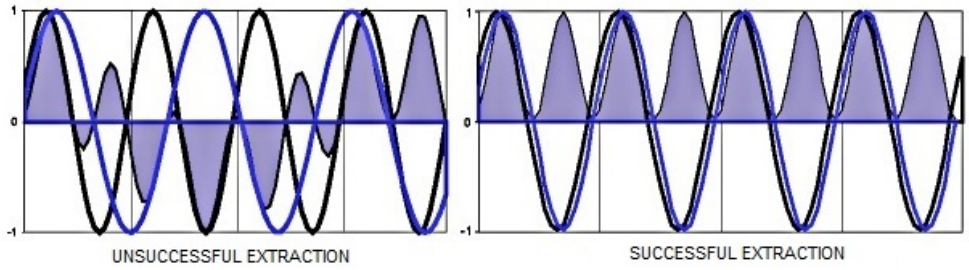


Image 4: An example of successful and unsuccessful extraction of frequency

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/(log2(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 x0, ..., xN-1 be complex numbers. We have already seen that DFT is defined by the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k \frac{n}{N}} \qquad k = 0, \ldots, N-1.$$

Evaluating this definition directly requires N2 operations: there are N outputs of Xk, 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 N2 complex multiplications and N(NÃ¢Ë†â€™1) complex additions. FFT algorithm can compute the same result with only (N/2)log2(N) complex multiplications and Nlog2(N) complex additions.

	DFT	**FFT**
complex multiplications	N2	(N/2)log2(N)
complex additions	N(N-1)	N/log2(N)

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 N2 to N log2(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]).

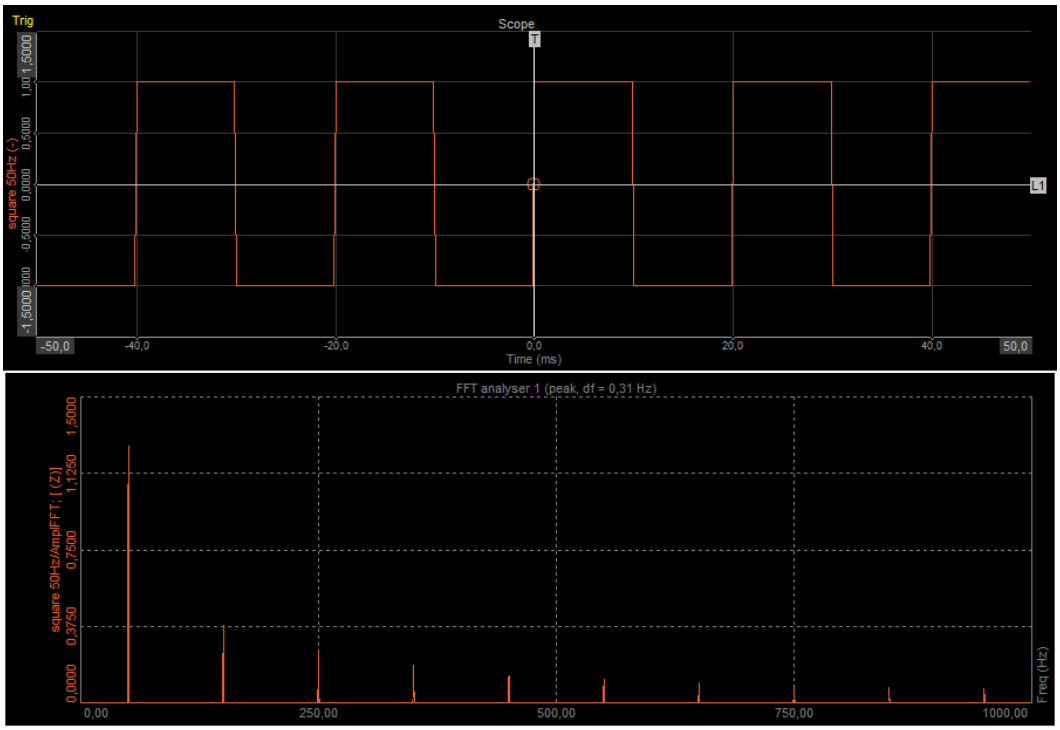


Image 5: Time and frequency representation of a square wave signal

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.

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:

$$SampleRate = MaximumSignalFrequency \cdot 2 \cdot 1.25$$

1.25 is the absolute minimum factor for getting the right values also in the upper region of the FFT. This is the equation another way around famous Nyquist criteria, which says that maximal signal frequency adequately presented in the digitized wave is the half of the sampling rate.

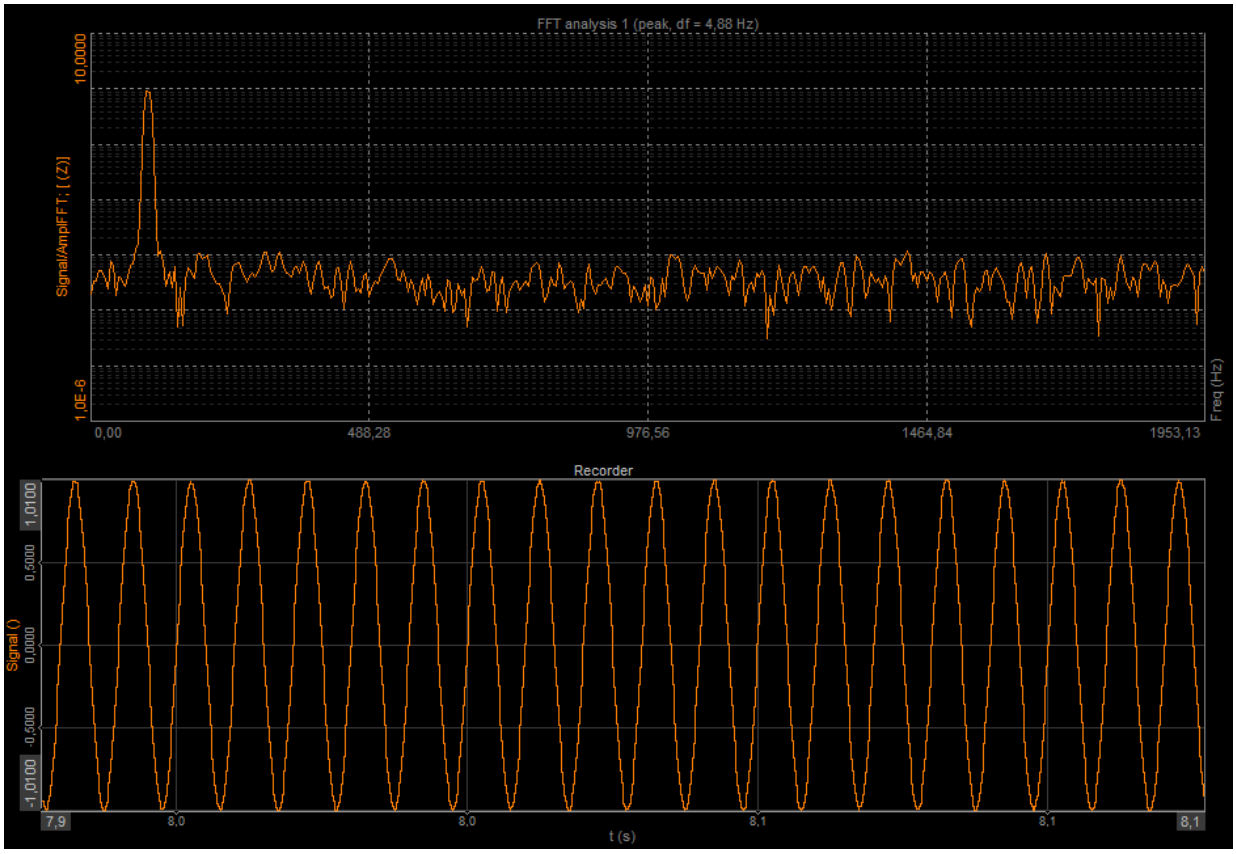


Image 6: Typical FFT screen

The result of FFT is a set of amplitudes of certain frequencies. The number of lines in the set is user-selectable, but they only change 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:

$$LineResolution = \frac{SampleRate/2}{NumberOfLines}$$

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 larger frequency lines it takes more time to calculate FFT.

$$TimeToCalculate = \frac{NumberOfLines \cdot 2}{SampleRate}$$

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

$$LineResolution = \frac{1}{TimeToCalculate}$$

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

Number of lines	Line resolution [Hz]	Calculation time [s]
512	21,5	0,046
1024	10,75	0,093
4096	2,685	0,372
16384	0,67	1,49

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.

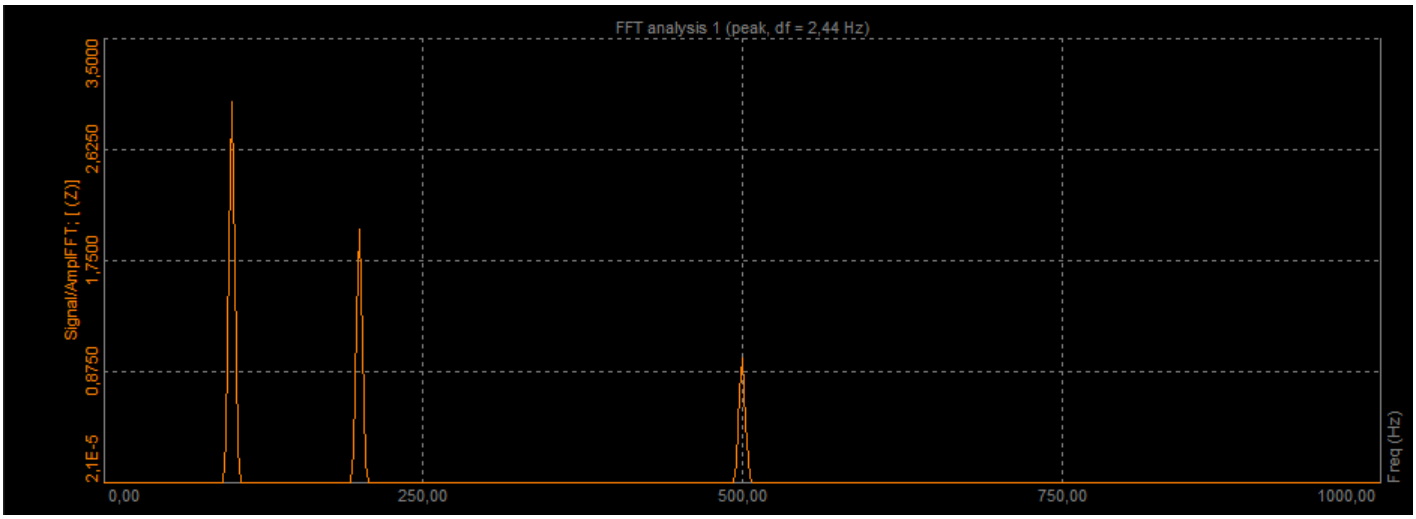


Image 7: FFT results displayed on a linear scale

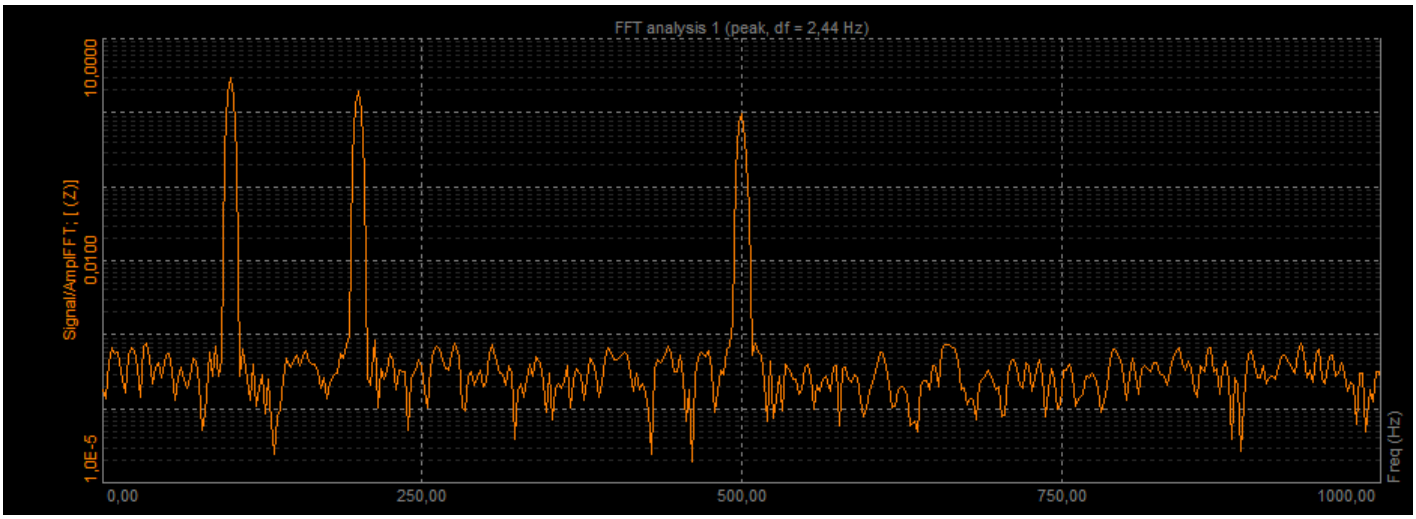


Image 8: FFT results displayed on a logarithmic scale

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.

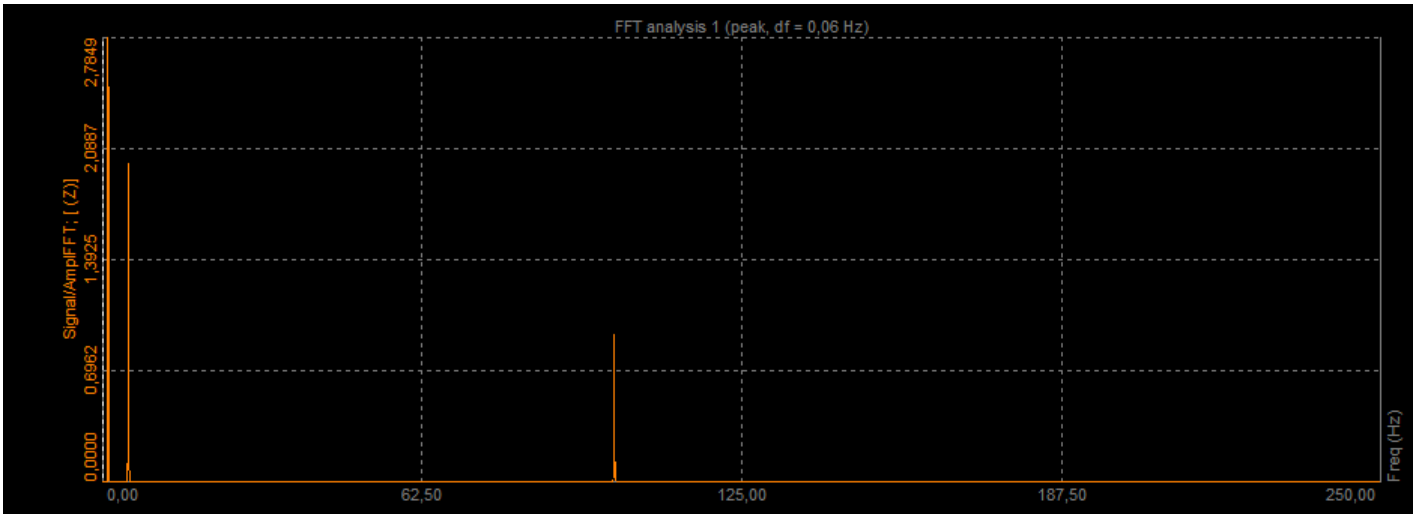


Image 9: Linear frequency scale

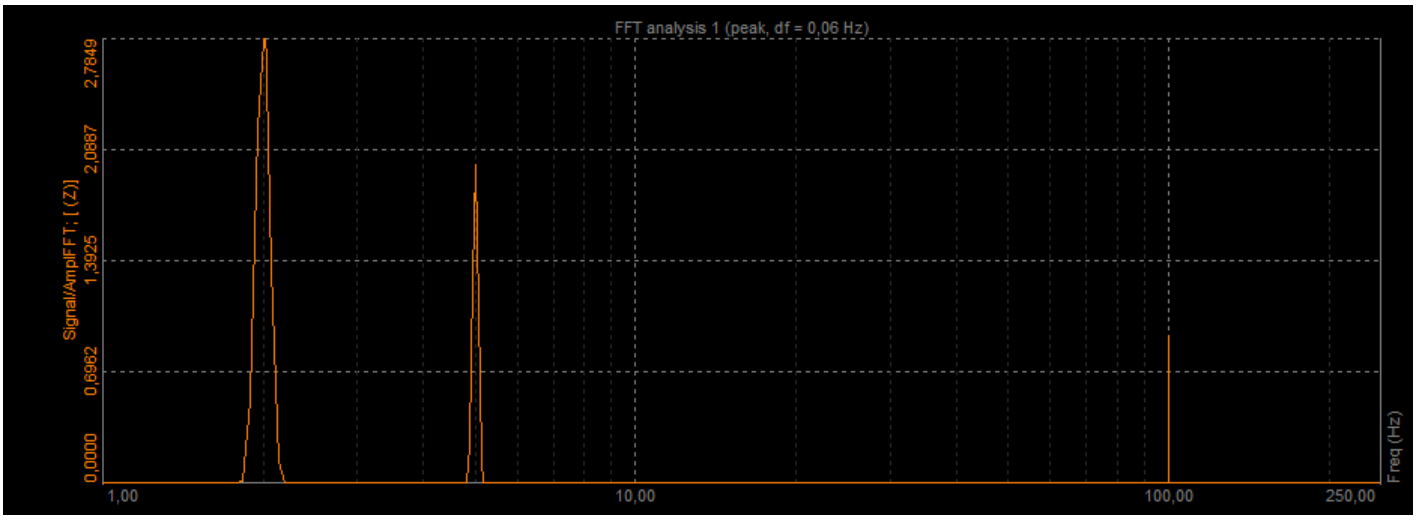


Image 10: Logarithmic frequency scale

If we use another technique, called CPB (constant percentage bandwidth) or 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 the sine wave is not on the frequency line, we get high amplitude values on both sides of the main band. The amplitudes are really high (with no window, it is 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 occurring close to larger ones.

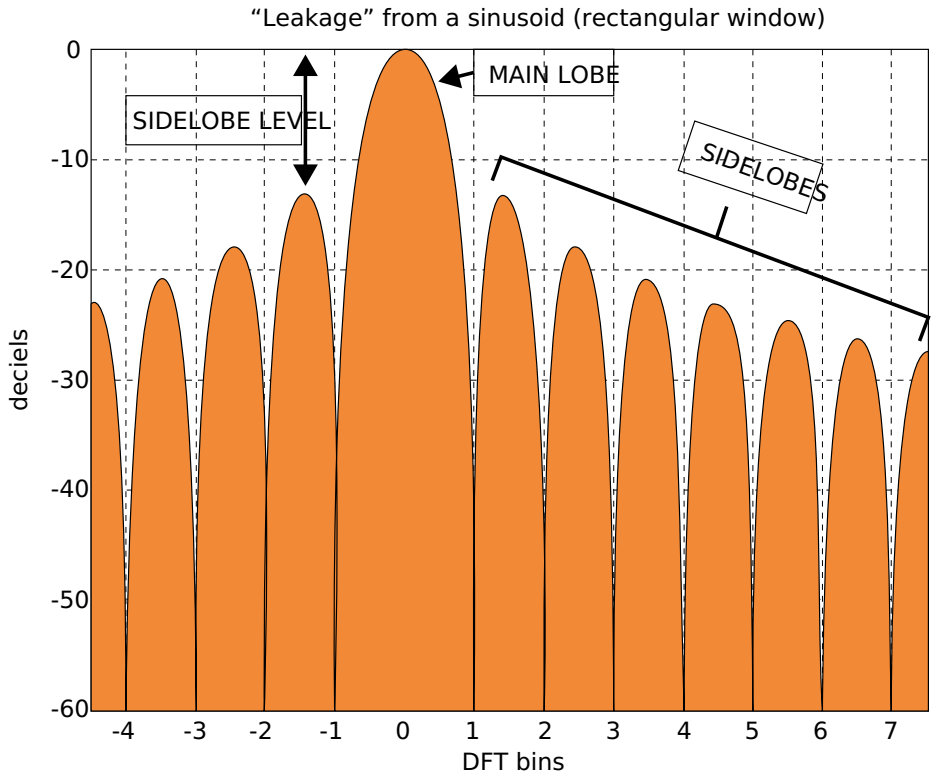


Image 11: Leakage from a sinusoid

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.

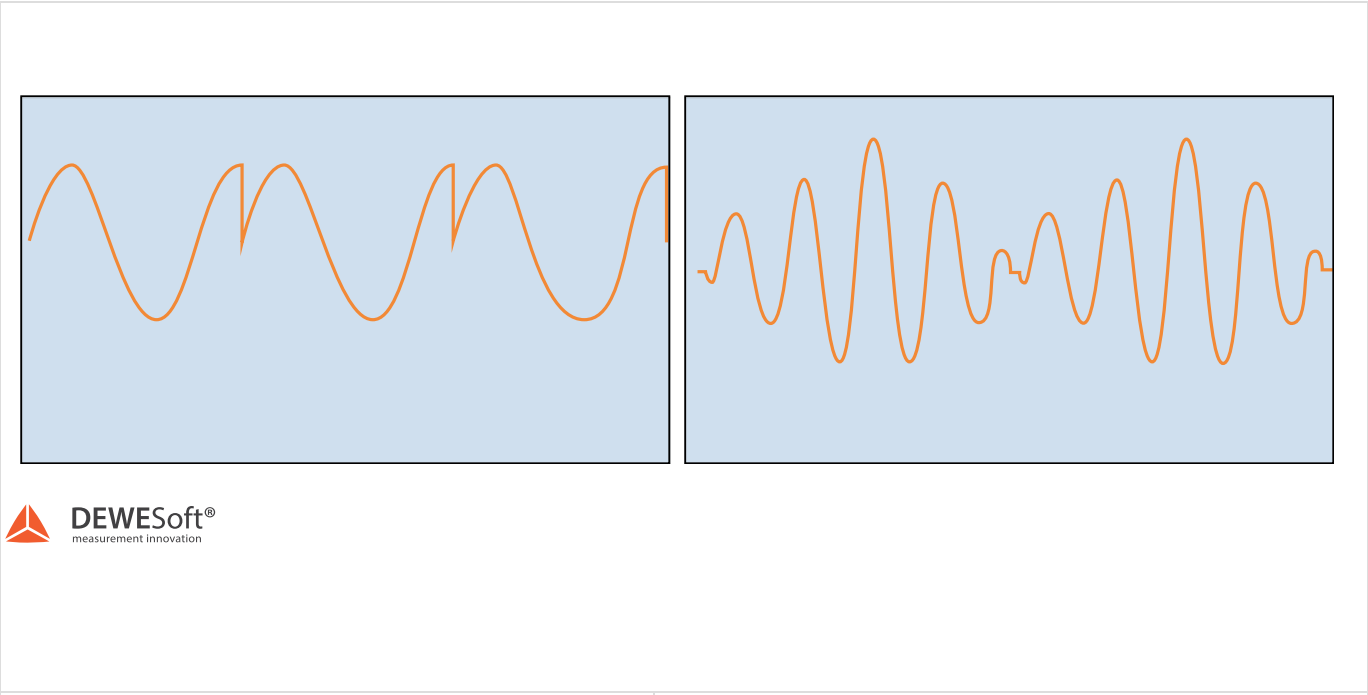


Image 12: Periodized signal with discontinuities	Image 13: Discontinuities "ironed out" by windowing

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

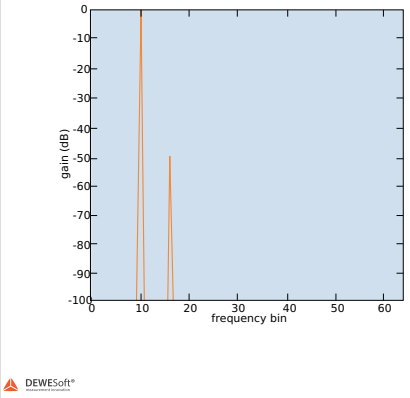


Image 14: Spectrum display without spectral leakage

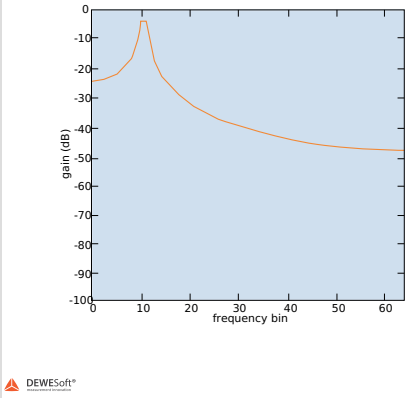


Image 15: Small frequency peak obscured as a result of leakage

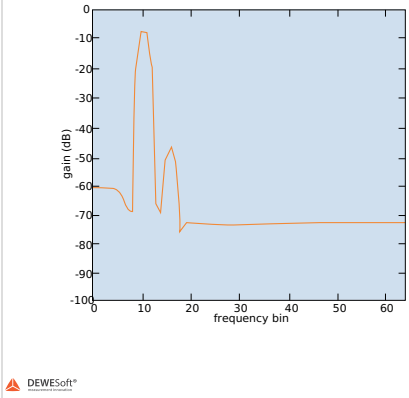


Image 16: Adjacent peak spectrum display with windowing, smaller frequency peak is no longer obscured.

Windows are characterized by a number of properties as shown in the picture below.

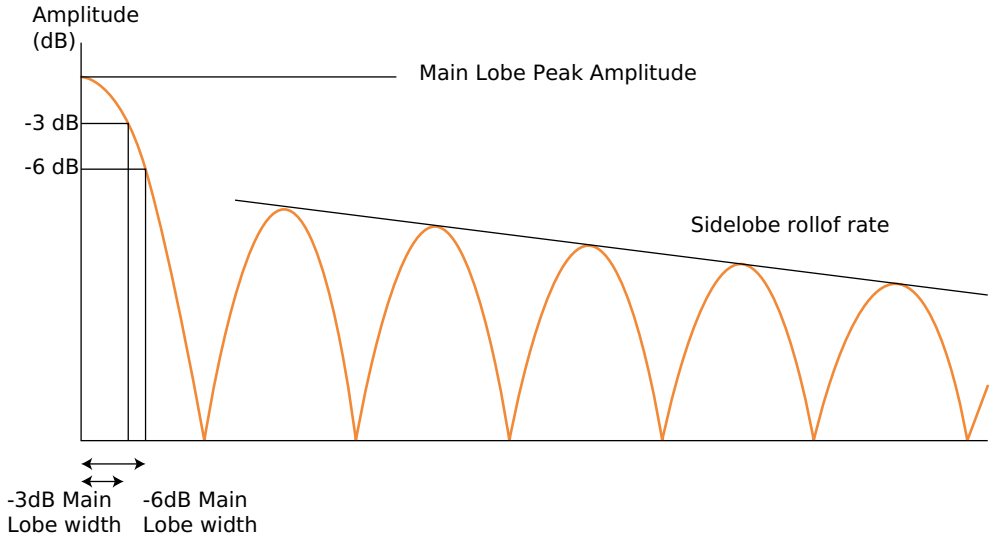


Image 17: Characterisation of windowing functions

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.

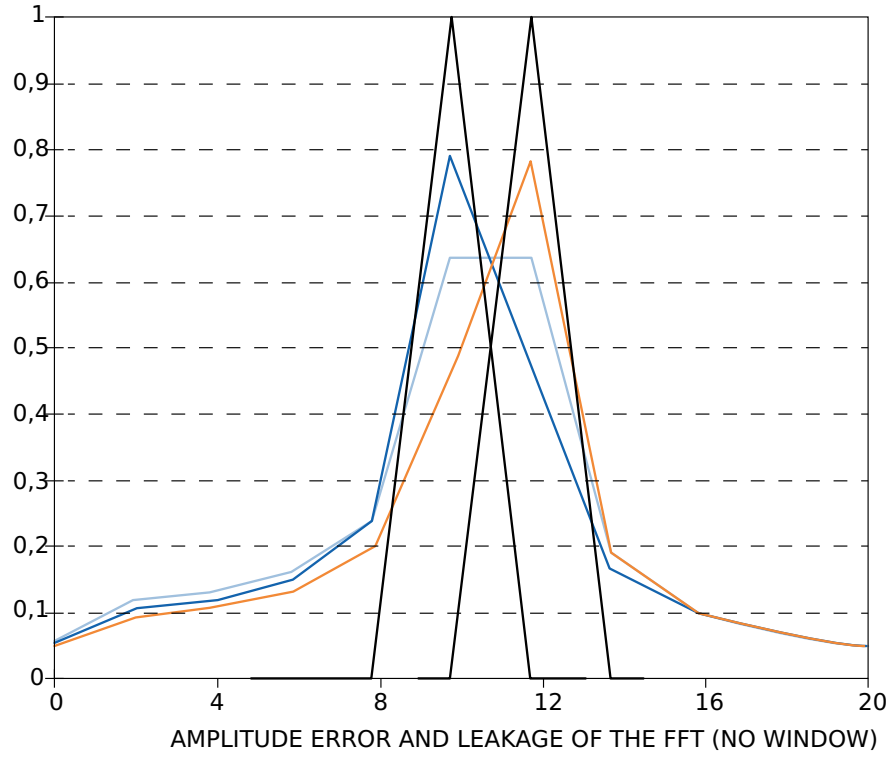
Fourier transformation errors

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)

The sum can produce "non-null' results even when the signal does not correspond to the frequencies extracted from the signal. The pure frequencies are because of that 'leaked' over neighbor frequencies. 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.



DEWESoft®
measurement innovation

Image 18: Amplitude error and leakage of the FFT result (no window)

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 original time-domain signal and because 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.

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

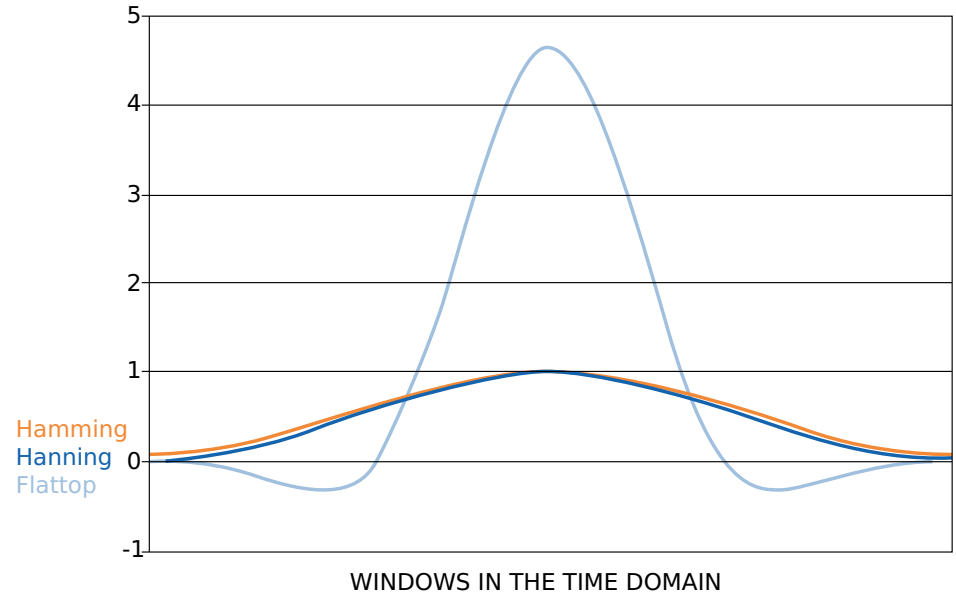


Image 19: Windowing functions in time domain

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: with modern equipment with lots of lines, this is no longer that much of a problem.

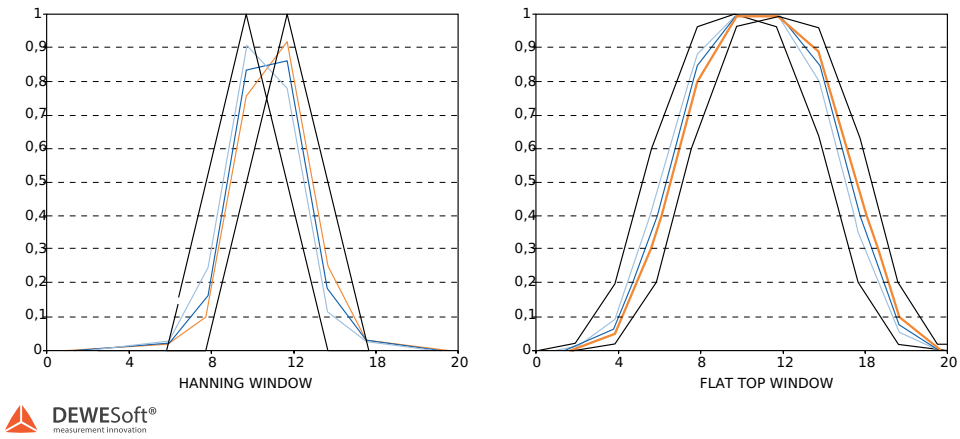


Image 20: Hanning window (left) and Flat top window (right)

Window characteristics (maximum amplitude error, sideband width, highest sideband attenuation, a 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.

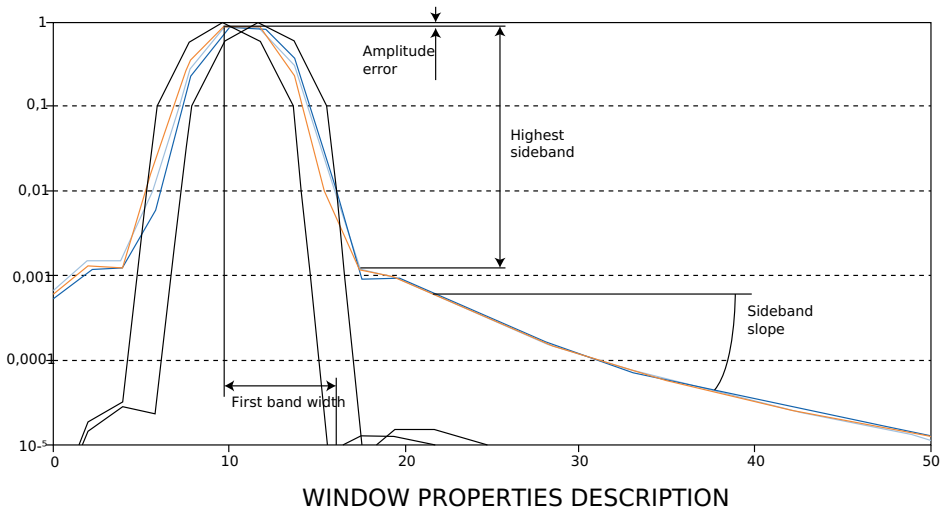


Image 21: Window properties description

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.

Window type	Maximum amplitude error [%]	Width of the first band [line]	Highest sideband [%]	Sideband slope [dB/decade]
Rectangular	36	2	22	-20
Hanning	15	2	2,5	-60
Hamming	18	2	0,7	-20
Blackman	12	3	0,12	-40
Flat top	0,02	5	0,04	-20

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.

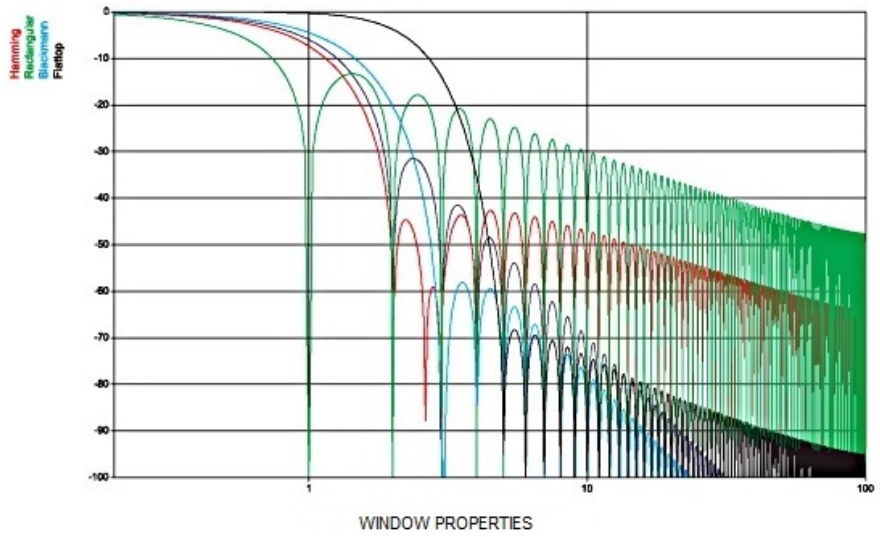


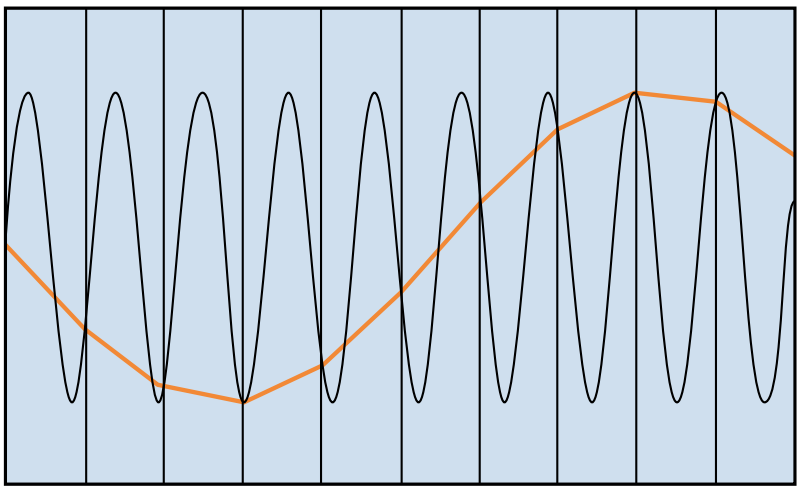
Image 22: Zoomed FFT of a pure sine wave

If a sine wave signal frequency falls between two lines, we see only the values of 0.5, 1.5, 2.5..., which always produce higher sidebands. 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 (50 Hz in Europe or higher), 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

The other problem comes from the fact of the signal conditioning. If simple A/D converters are used, the sampling frequency must be at least twice higher than the maximum frequency of the signal. This is called the Nyquist theorem (Aliasing effect). 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.



ALIASING

DEWESoft®
measurement innovation

Image 23: Aliasing effect in the time domain

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 aliases. 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) with no anti-aliasing filter are used and the

online FFT analyser 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 upper right side, there is a picture of a function generator, with the output frequency in red rectangular.

The first output frequency from a function generator was 400 Hz. Also, the frequency detected by our hardware was 400 Hz.

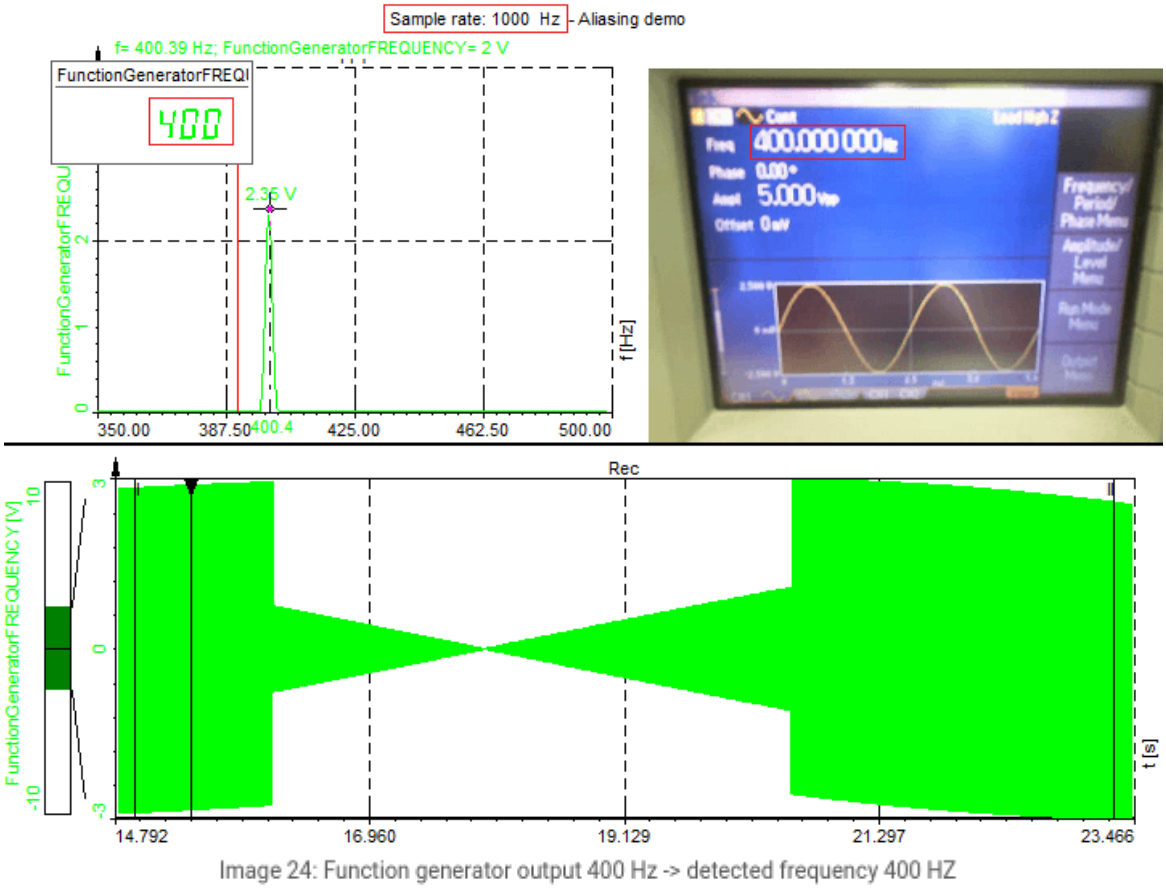


Image 24: Function generator output 400 Hz -> detected frequency 400 HZ

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 of 0 Hz. This is because of the Nyquist theorem, which is described above.

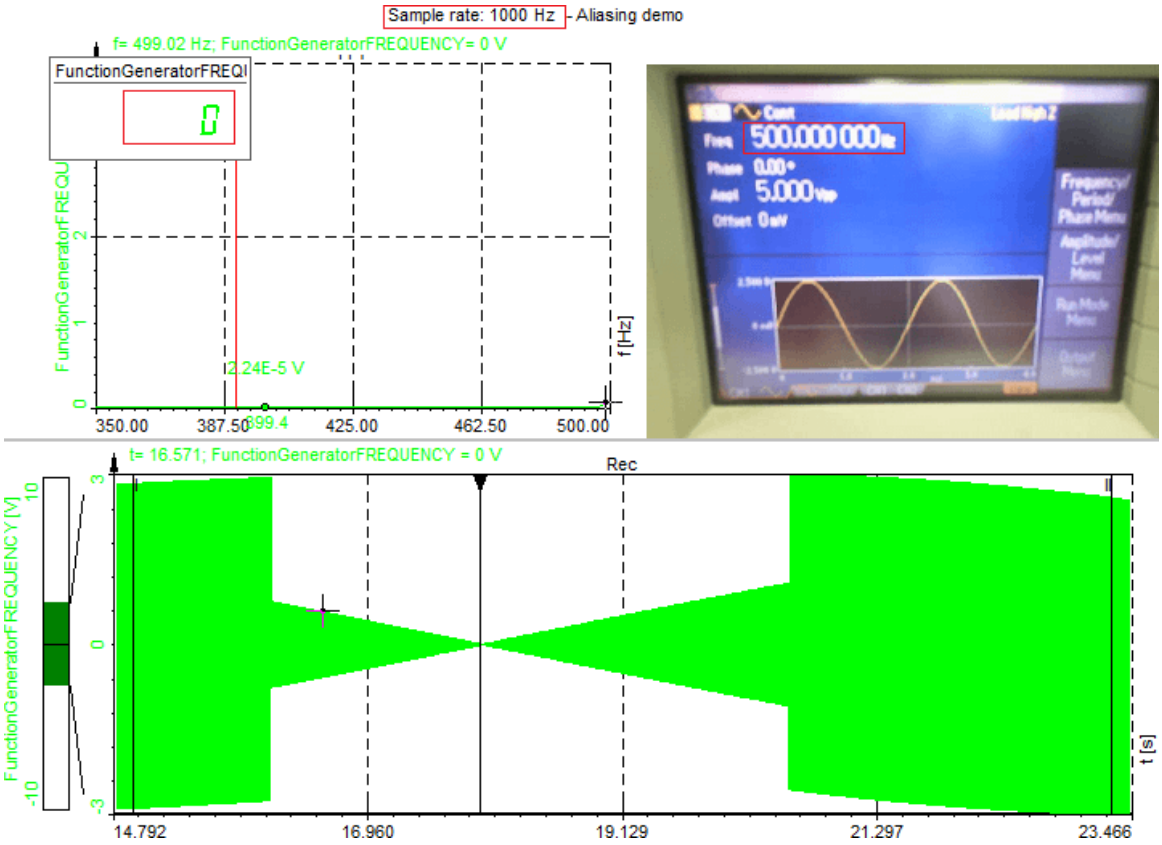


Image 25: Function generator output 500 Hz -> detected frequency 0 Hz

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.

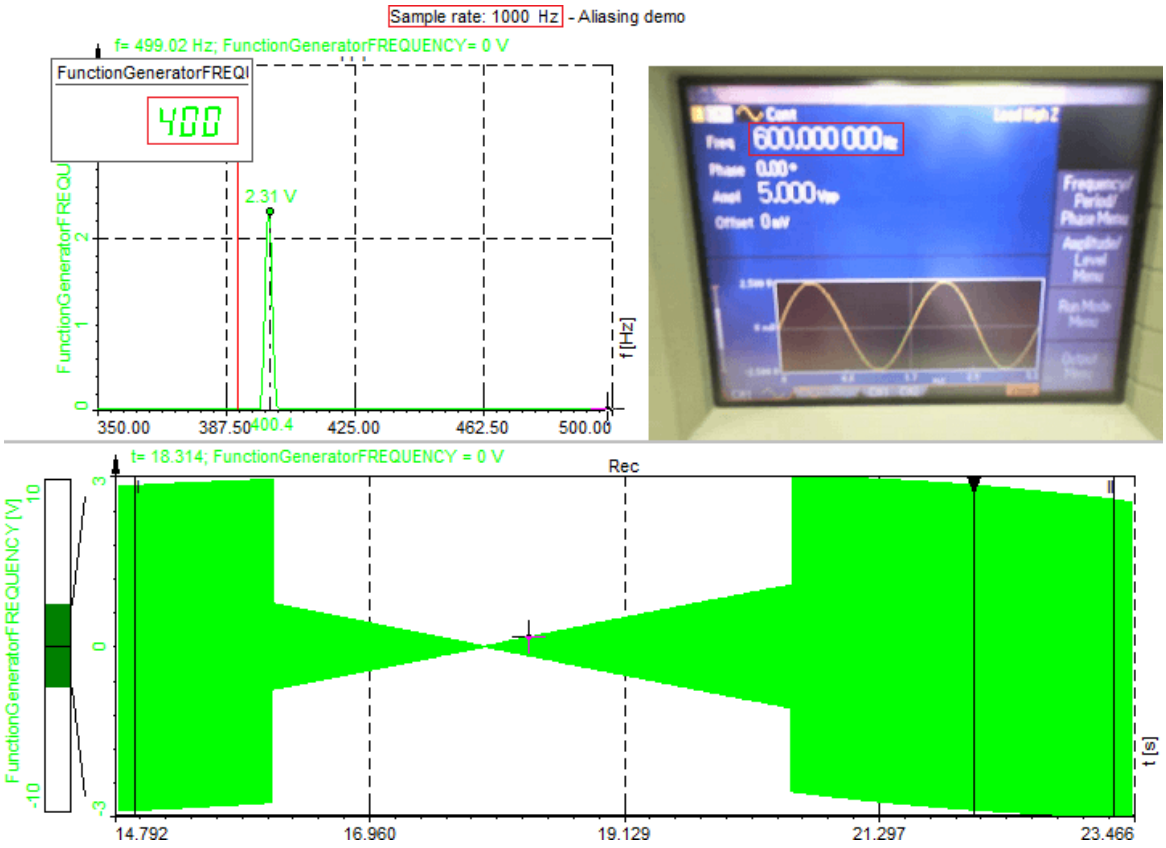


Image 26: Function generator output 600 Hz -> detected frequency 400 Hz

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 many FFTs during the time and average frequency lines.

There are many ways to average the signal, but the most important are linear, exponential and peak hold average.

- linear averaging - each FFT counts the same in the results
- exponential averaging - FFTs becomes less and less important with time
- peak hold average - only maximum results are stored and shown

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.

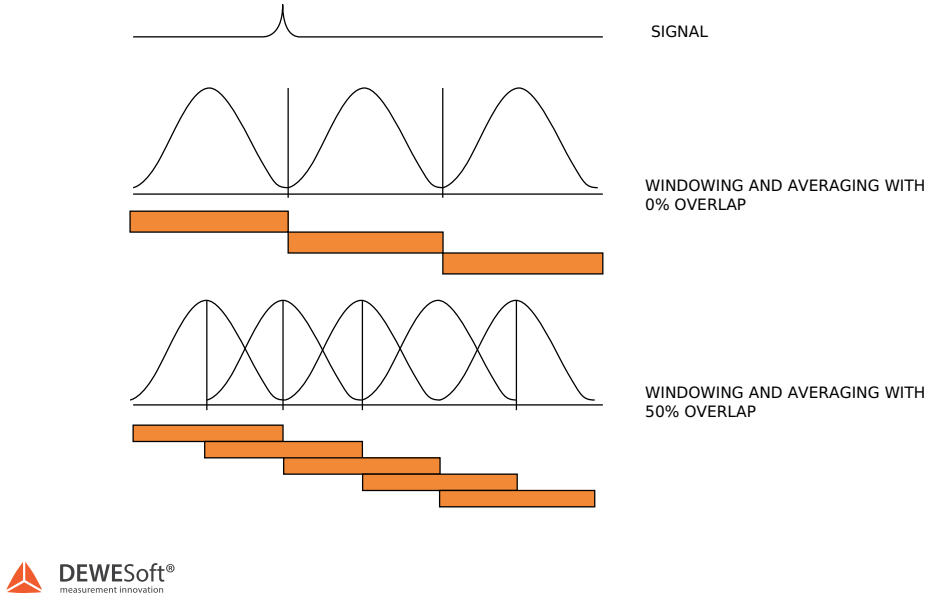


Image 27: Graphical representation of overlap function

In the image above, 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.

Overlapping

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. More the signal is not like a sine, the higher the harmonics are.

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,. . .. 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.

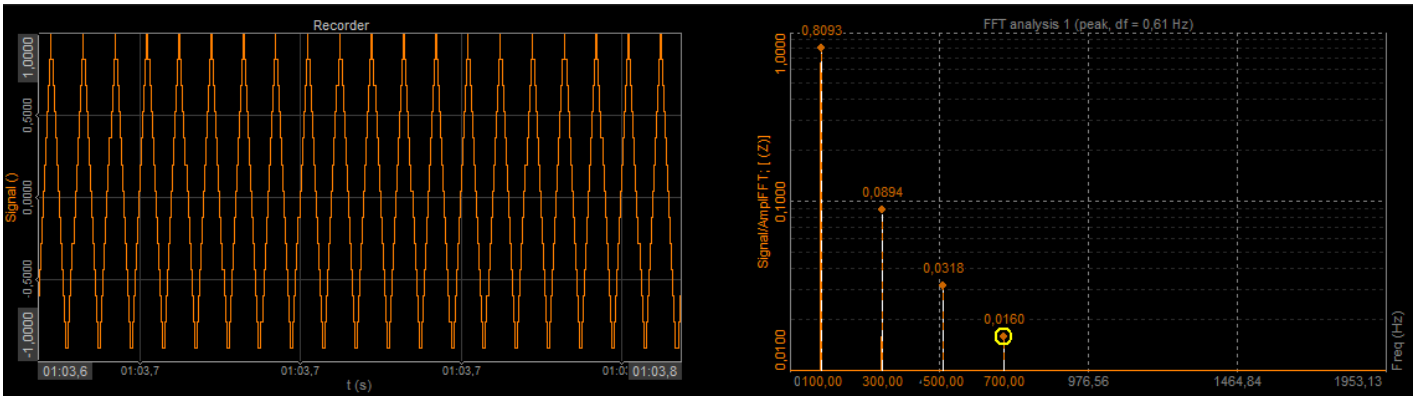


Image 28: Triangle signal in time and in the frequency domain

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.

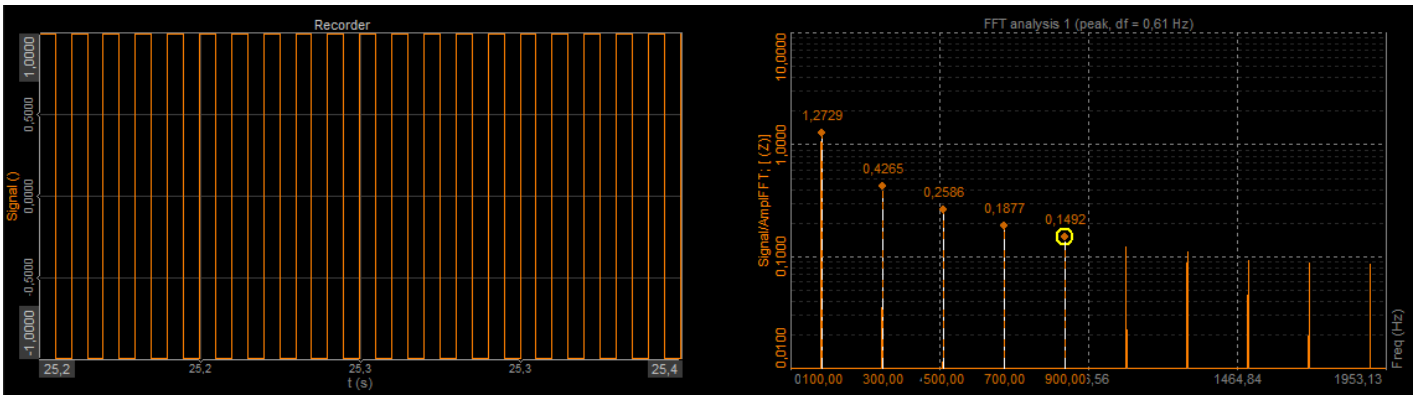


Image 29: Rectangular (square) signal in time and in the frequency domain

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. The other ones are swept sine and noise, but this is already a part of another story - dual-channel frequency analysis.

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.

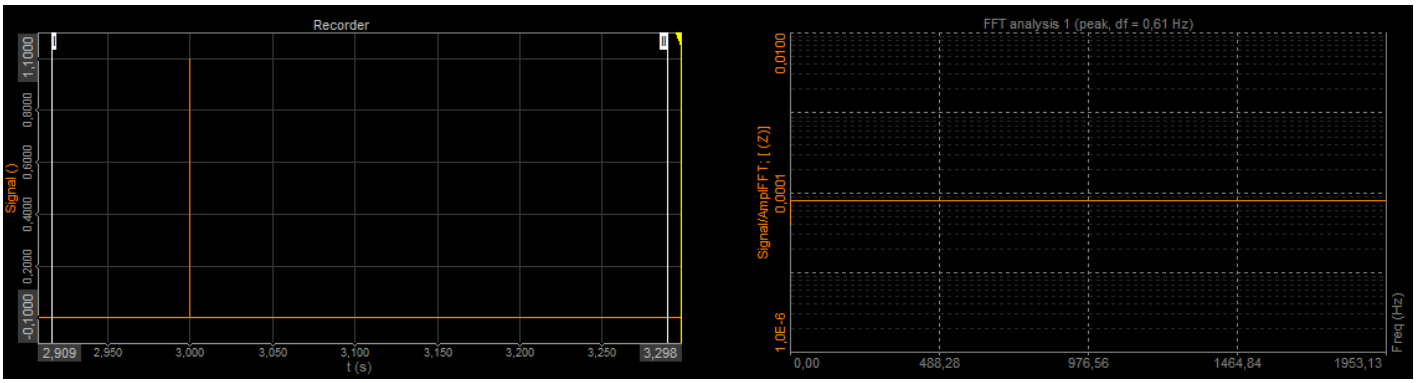


Image 30: Impulse signal in time and in the frequency domain

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.

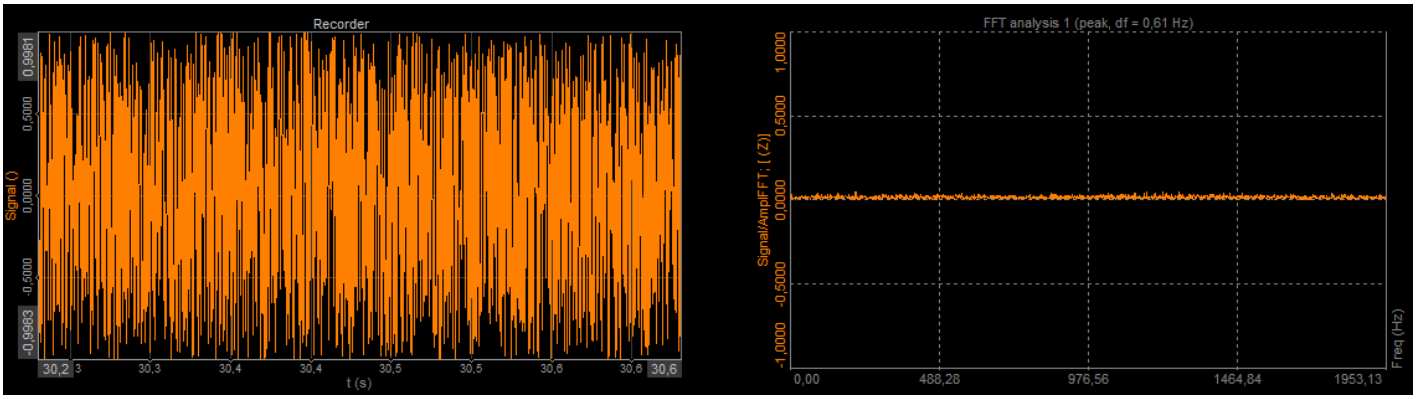


Image 31: White noise signal in time and in the frequency domain

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 NBeating signal in the frequency domain.

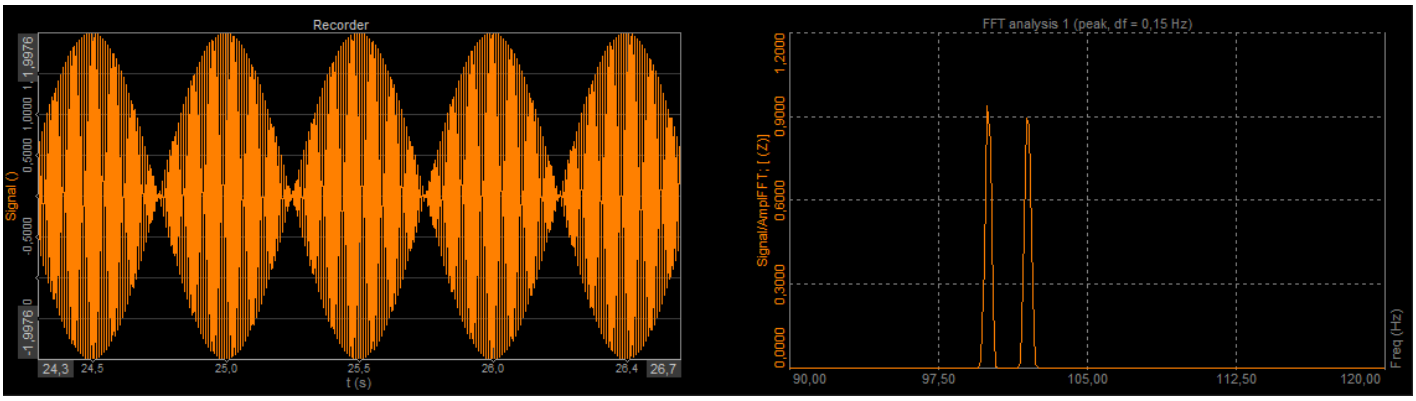


Image 32: Beating signal in time and in the frequency domain

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, which is the base for the FFT calculation, should show some modulation peaks. When windowing is used (we know that the baseband could be even 4 lines wide) and the main band, which is always the highest, covers the modulation with low line resolution, time signal should show at least 16 or 32 modulation peaks that the modulation is shown in the FFT.

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.

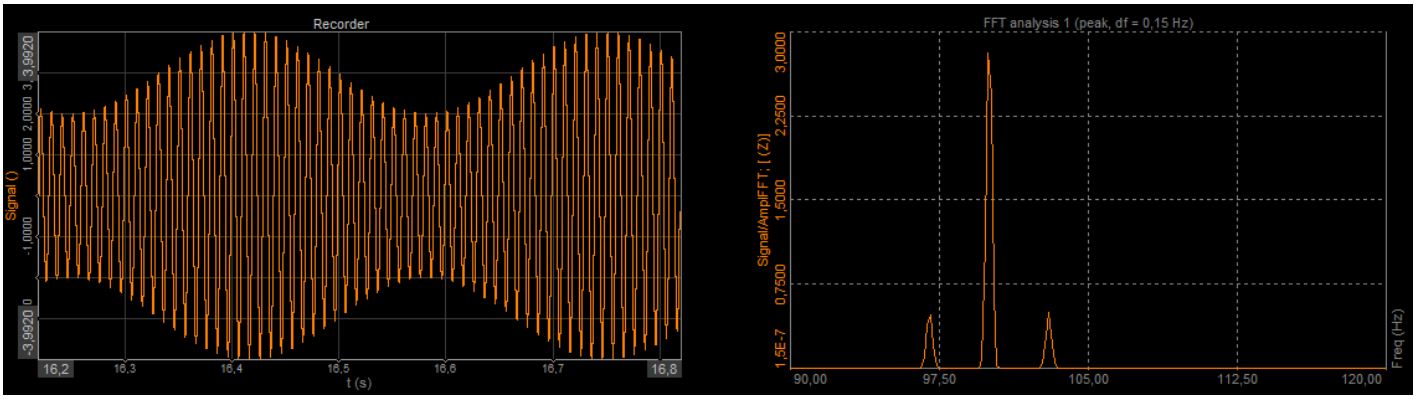


Image 33: Amplitude modulated signal in time and in the frequency domain

FFT analysis module in Dewesoft

In Dewesoft, we add a new FFT analysis module by selecting the + button and then the FFT Analyser.

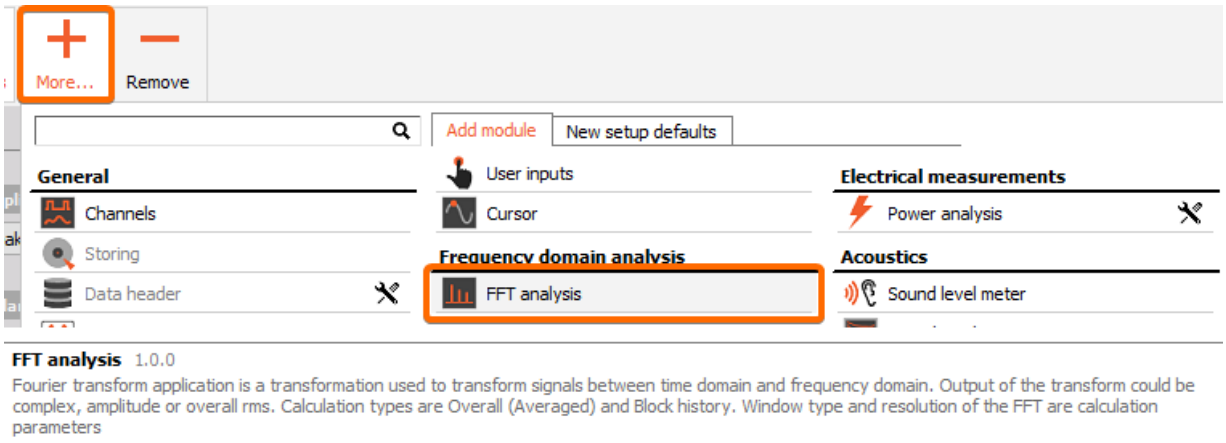


Image 34: Adding a new FFT analysis module in Dewesoft

When we add a new FFT analysis module the following setup appears:

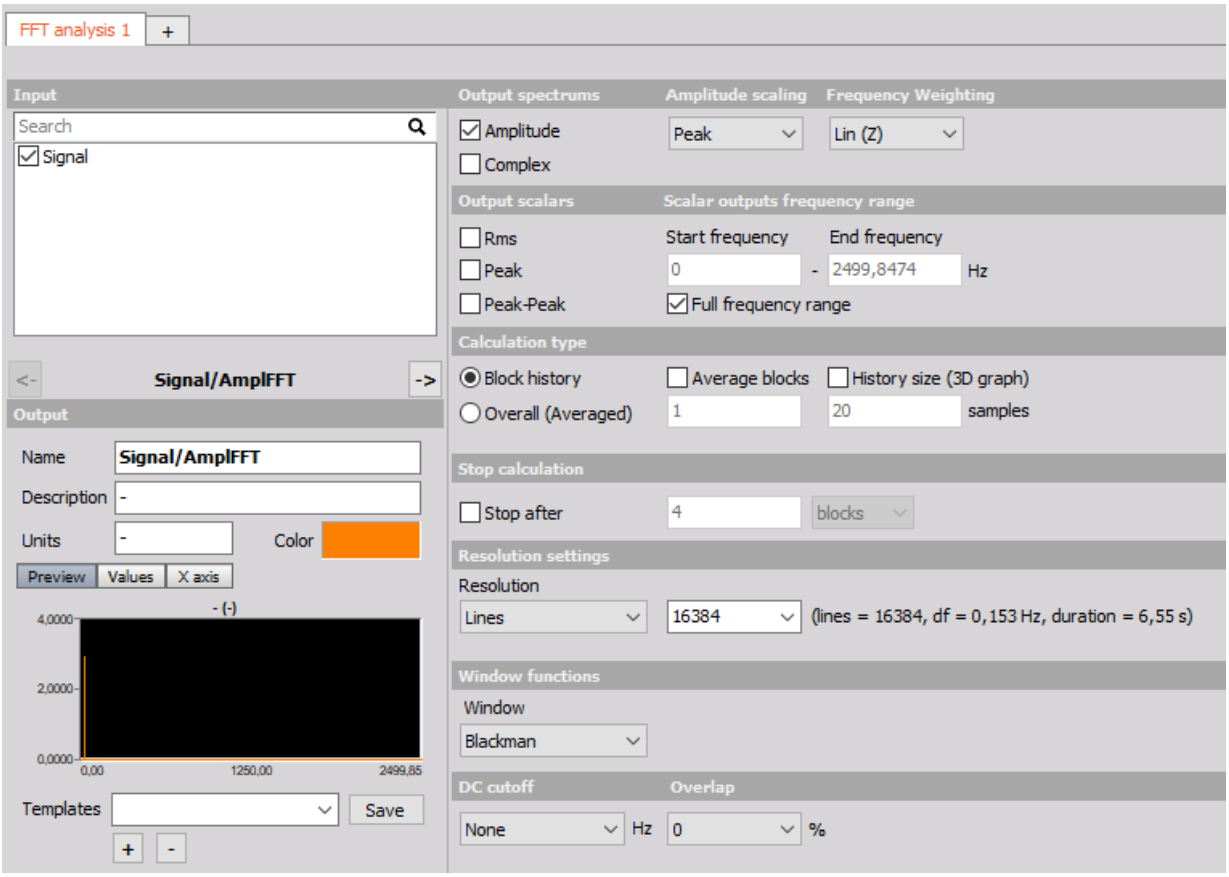


Image 35: FFT analysis module setup in Dewesoft

Output spectrums

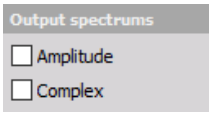


Image 36: Amplitude and Complex output channels

- Complex - outputs are phase, imaginary and real part of the signal
- Amplitude - output is the amplitude of the signal

Calculation type

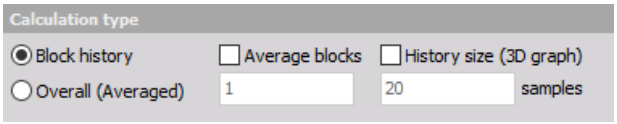


Image 37: Different possibilities for calculation type

Block history calculation type uses blocks to calculate the FFT spectrum.

For example, let's set the FFT resolution to 1024 lines.

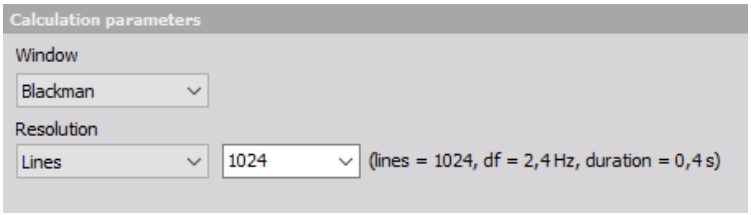


Image 38: Setting the windowing function and line resolution

The FFTs are acquired shot by shot and put into the buffer. For every 1024 lines, a new FFT will be calculated and shown on the 2D graph.

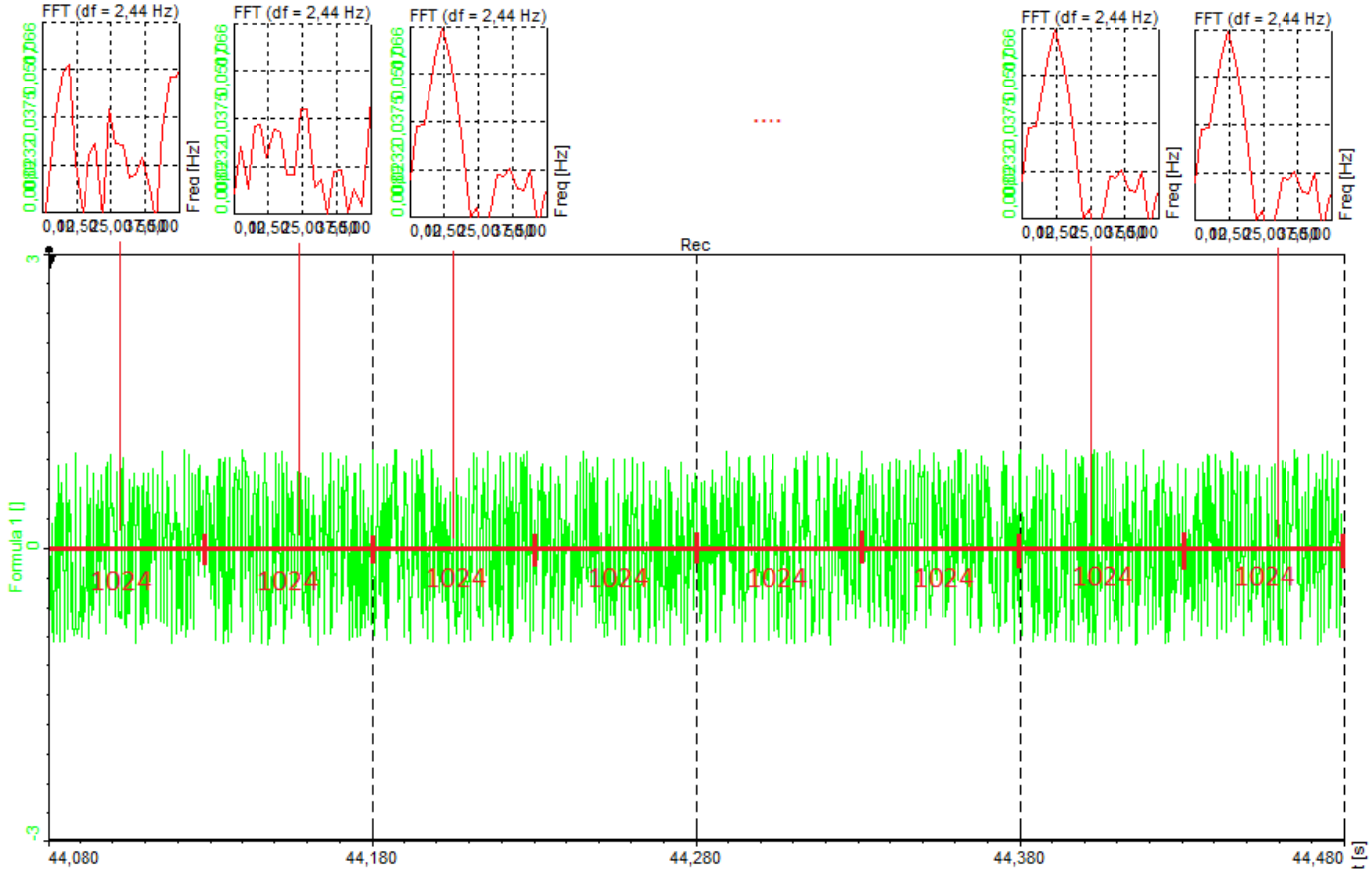


Image 39: Block history calculation - FFT is calculated for each block of data individually

With the block history calculation type, we can average more blocks together.

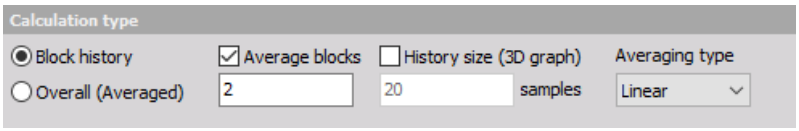


Image 40: Averaging of data blocks

Let's say we want to average 2 blocks of the signal. Each new FFT will be calculated for two blocks together.

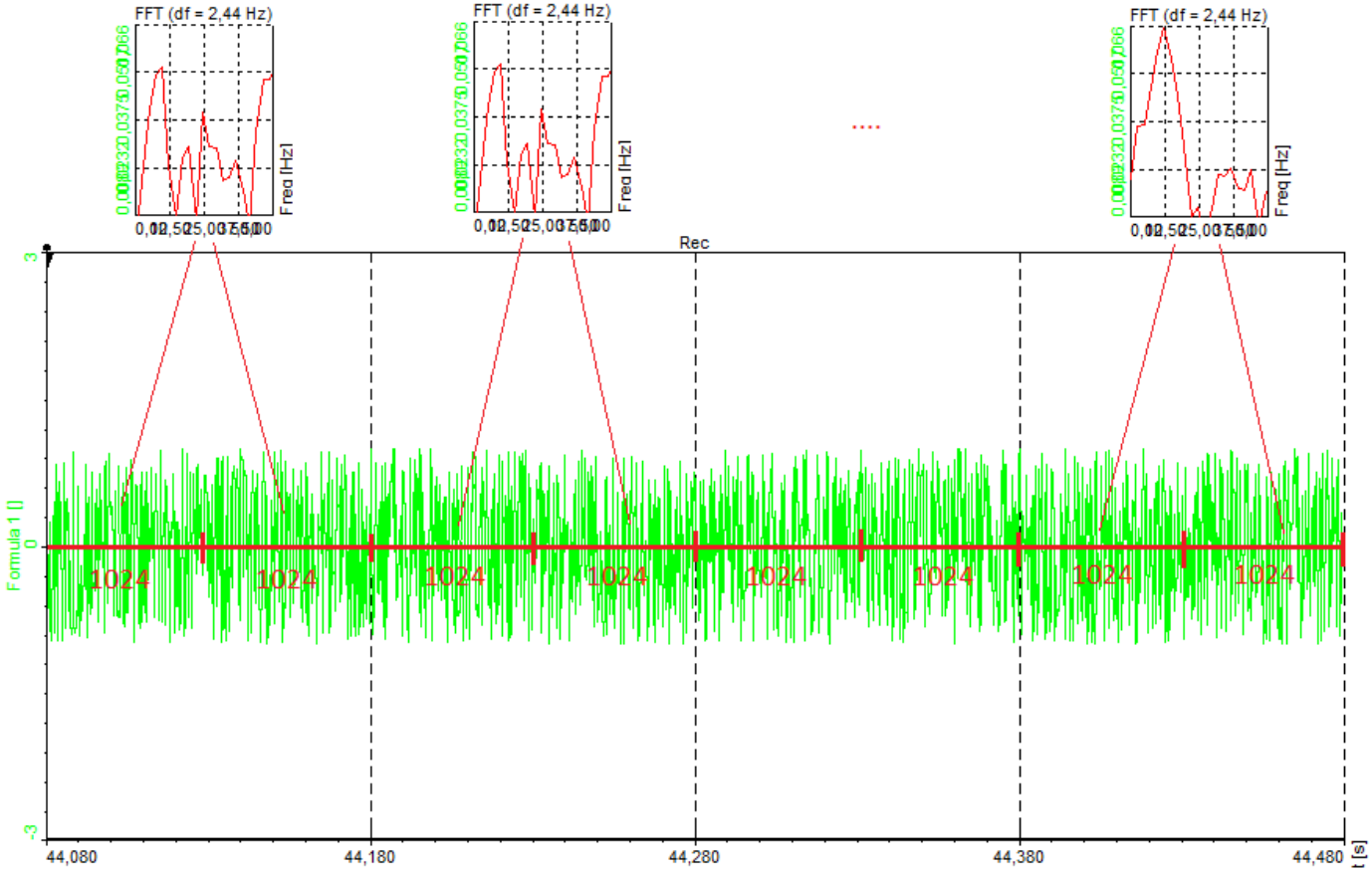


Image 41: FFT is averaged from 2 blocks of data

FFT spectrum can also be observed on the 3D graph.

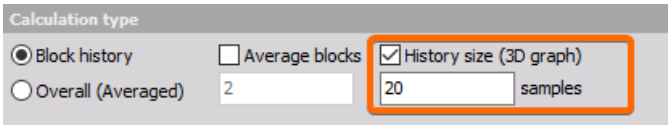


Image 42: History size (3D graph) option for displaying the FFT results on the 3D graph (FFT vs. time)

Overall (Averaged) calculation type gives only one averaged FFT spectrum at the end of the measurement. It will average all the blocks in the signal and the output will be only one FFT for the whole measurement.

Image 43: Overall (Averaged) calculation type

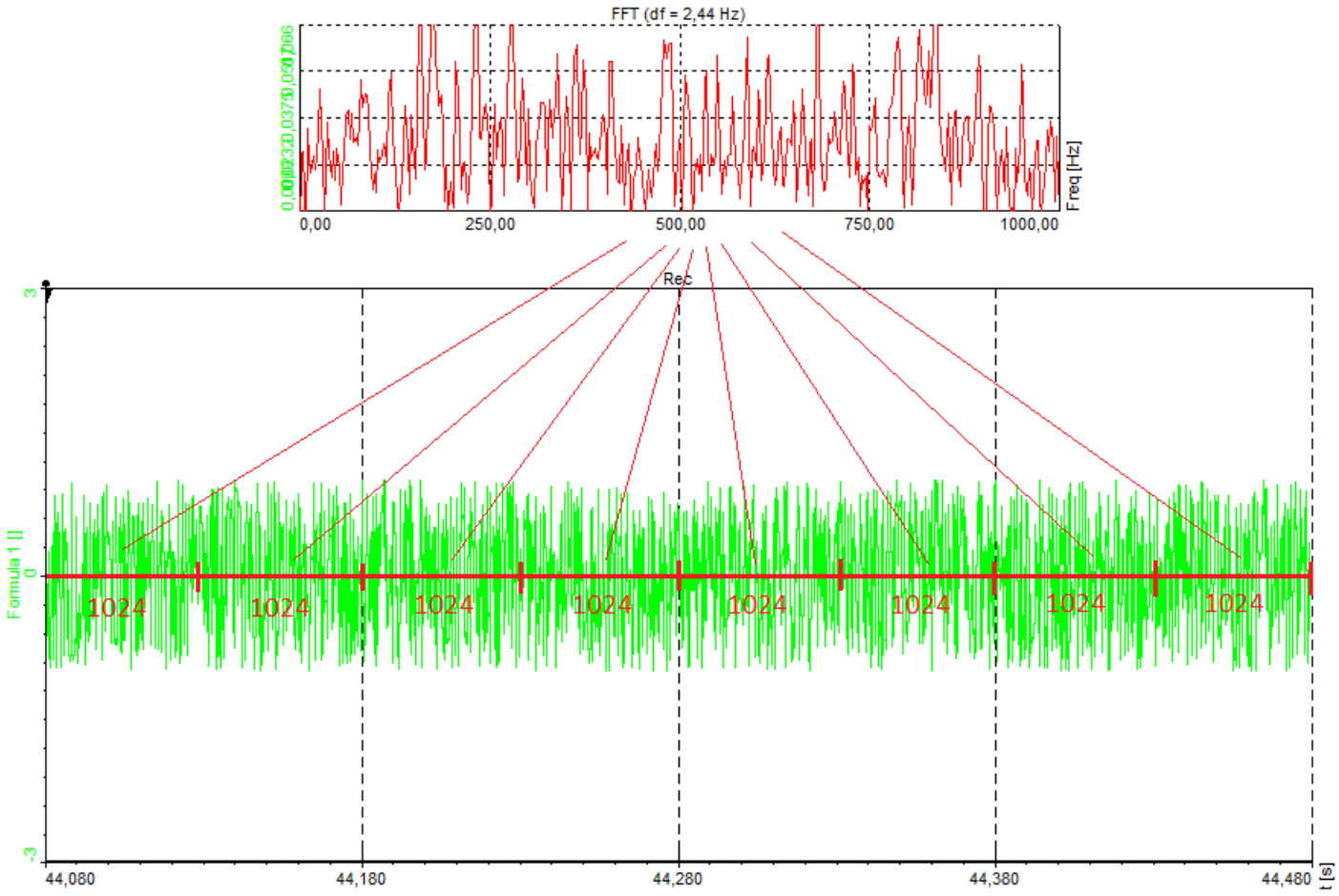


Image 44: With Overall (Averaged) option we acquire one FFT for the whole measurement

Calculation parameters

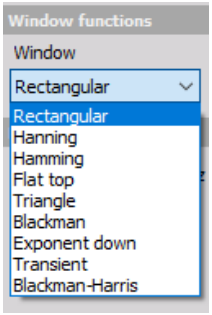


Image 45: Windowing functions

Window functions were already described on previous pages.

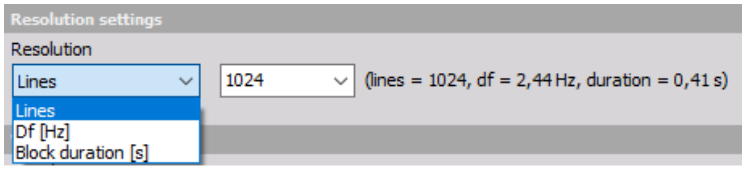


Image 46: Resolution settings

Resolution can be defined in the number of Lines or with the delta frequency Df (Hz).

The FFT lines are responsible for the frequency resolution. The higher the FFT lines value, the better the resolution - but also longer calculation time.

The line resolution depends on the sampling rate and the number of lines chosen for the FFT. So if we want to have fast response on the FFT, we choose fewer lines, but we will have a lower frequency resolution. If we want to see the exact frequency, we set a higher line resolution. The simple rule is: if it takes 1 second to acquire the data from which the FFT is calculated, the resulting FFT will have a 1 Hz line resolution. If we acquire data for 2 seconds then the line resolution will be 0.5 Hz.

Example: The sampling rate has been set to 10000 samples/sec and the resolution of 1024 FFT lines. These settings allow an FFT analysis of up to 5000 Hz (half the sampling rate). Now you divide the max analyses frequency by the FFT lines (5000 Hz / 1024 lines). The result is 4.88 Hz per line resolution (mentioned in the selection line).

Amplitude type

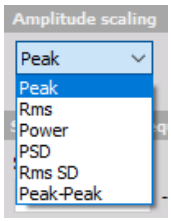


Image 47: Amplitude scaling

The Amplitude type display section defines display in the Y-amplitude axis.

From the Amplitude scaling type display drop-down list, we can select different types of amplitude scaling of the FFT. The basic setting is Amplitude (Auto), which shows for pure sine wave the amplitude of the sine.

- Real - is the pure signal amplitude [V]
- RMS - is the RMS amplitude, calculated as Amplitude/sqrt(2) [V]
- Power - calculated as RMS value squared [V*V]
- PSD - calculated as RMS squared, divided by the line resolution and sqrt(2) - used for checking the noise [V*V/Hz]
- RMS SD - calculated as RMS value, divided by the square root of line resolution - also used for checking the noise [V/sqrt(Hz)]
- Peak-peak - is the difference in amplitude from signals negative peak and positive peak [V]

DC cutoff

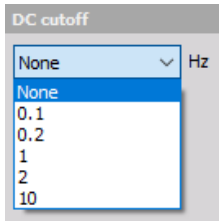


Image 48: DC cutoff options

To remove DC or low-frequency components, select from a drop-down list the DC cutoff filter - lower limit.

Overlap

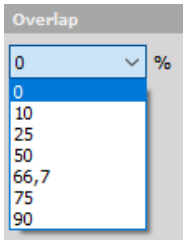


Image 49: Overlap selection

Overlap defines the percentage of time signal that has already been calculated and it is used again for calculation (example: 50% overlapping means that the calculation will take half of the old data).

When the window type is used, we have to use an overlap otherwise some of the data will be ignored. Therefore, the use of overlap is highly recommended.

Weighting

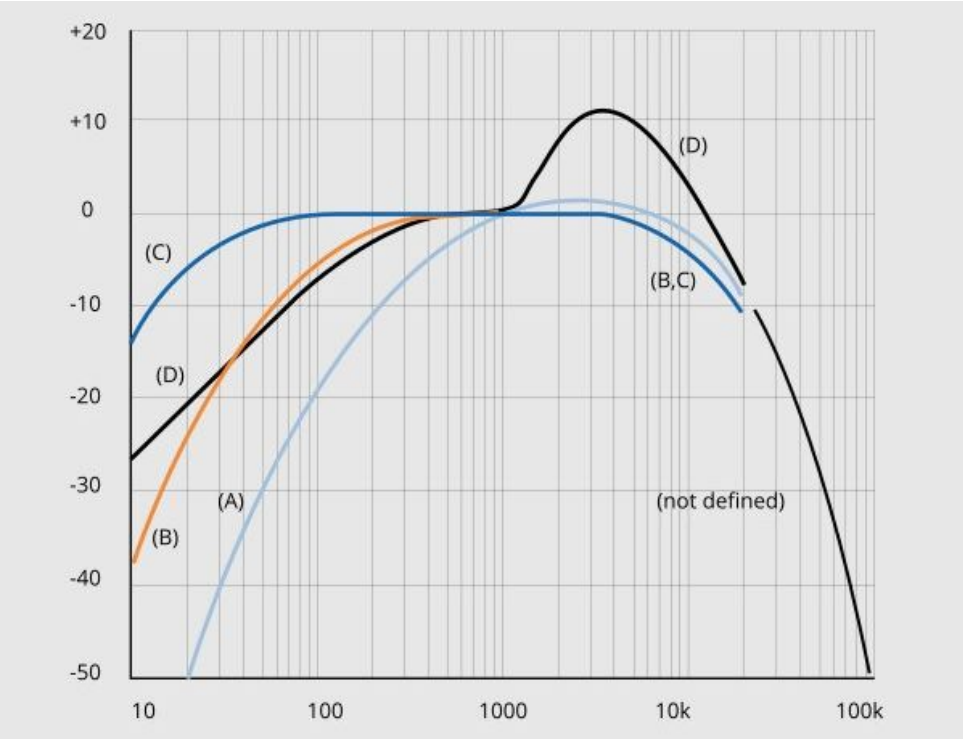


Image 50: Weighting curves

As a standard, FFT analyzer uses Linear Weighting. For sound analysis, special FFT weighting can be set. As opposed to the sound module in math, where the weightings will be calculated in the time domain, this will calculate the sound weighting in the frequency domain.

- **Linear weighting** - is linear at all frequencies and it has the same effect on all measured values.
- **A weighting** - A-weighting is applied to instrument-measured sound levels in an effort to account for the relative loudness perceived by the human ear, as the ear is less sensitive to low audio frequencies.
- **B weighting** - B-weighting is used for intermediate levels and is similar to A, except for the fact that low-frequency attenuation is a lot less extreme though still significant (-10 dB at 60 Hz). This is the best weighting to use for musical listening purposes.
- **C weighting** - C-weighting is similar to A and B as far as the high frequencies are concerned. In the low-frequency range, it hardly provides attenuation. This weighting is used for high-level noise.
- **D weighting** - D-weighting was specifically designed for use when measuring high-level aircraft noise. The large peak in the D-weighting curve is not a feature of the equal-loudness contours but reflects the fact that humans hear random noise differently from pure tones, an effect that is particularly pronounced around6 kHz

Averaging type

If we chose the Calculation type as Overall (Averaged), we have to select also the Averaging type.

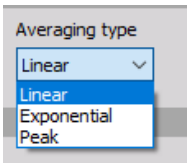


Image 51:
Averaging type

- linear averaging - each FFT counts the same in the results
- exponential averaging - FFTs becomes less and less important with time
- peak hold average - only maximum results are stored and shown

FFT visual control

There is another option on how to get an FFT on a signal. During the measurement add an FFT preview widget. Click the Design button and then add an FFT preview widget by clicking on the icon.

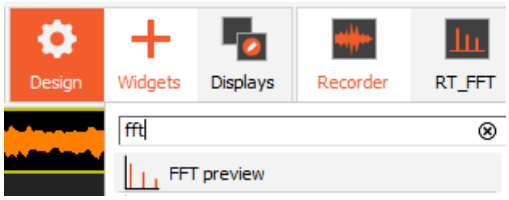


Image 52: Adding FFT preview

The FFT visual control can display the position and amplitude of maximum peaks, RMS values, or marked peaks.

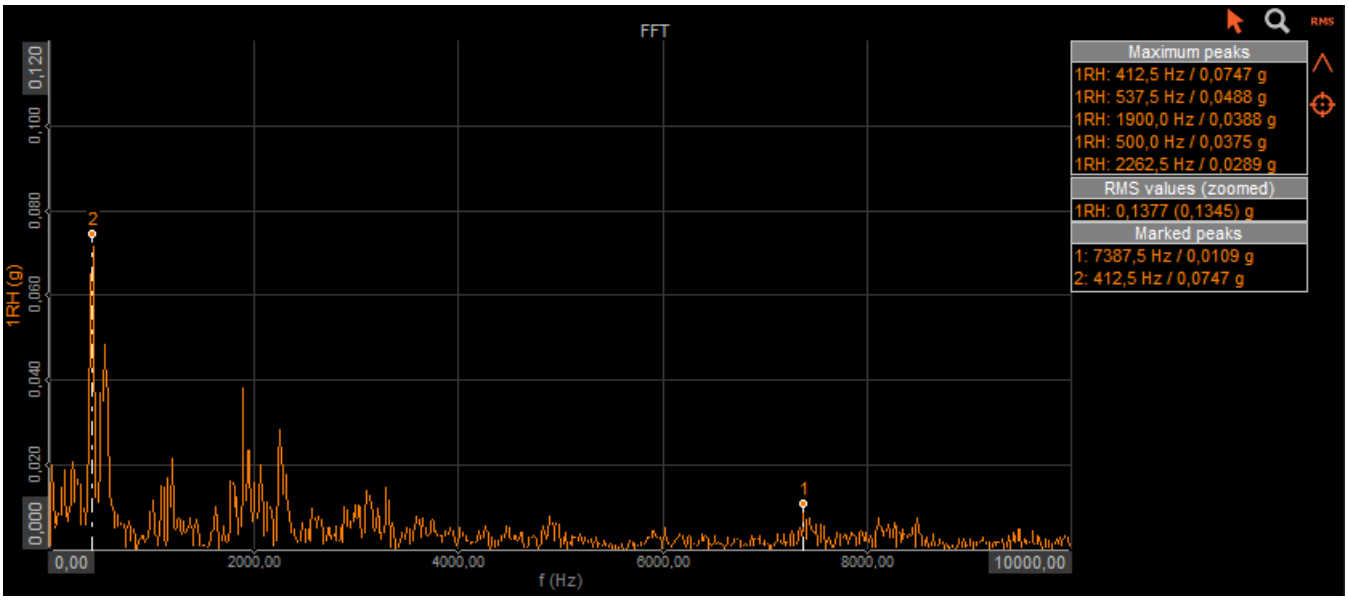


Image 53: FFT visual control in Dewesoft

Difference between FFT analysis module and FFT visual control

How FFTs collect data:

- **Visual FFT** always takes the values left and right in an equal amount from a position of the yellow cursor.
- **FFT analysis module** takes the values in the block from where you can see the timestamps. The start is on the first stamp and the end is on the second stamp.

Visual FFT is more dynamic to get quick look wherever you put yellow cursor and Math FFT gives you exact block so you know from where to where some FFT is exactly calculated.

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.

Image 54: Demo device with electromotor, accelerometer, and encoder

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.

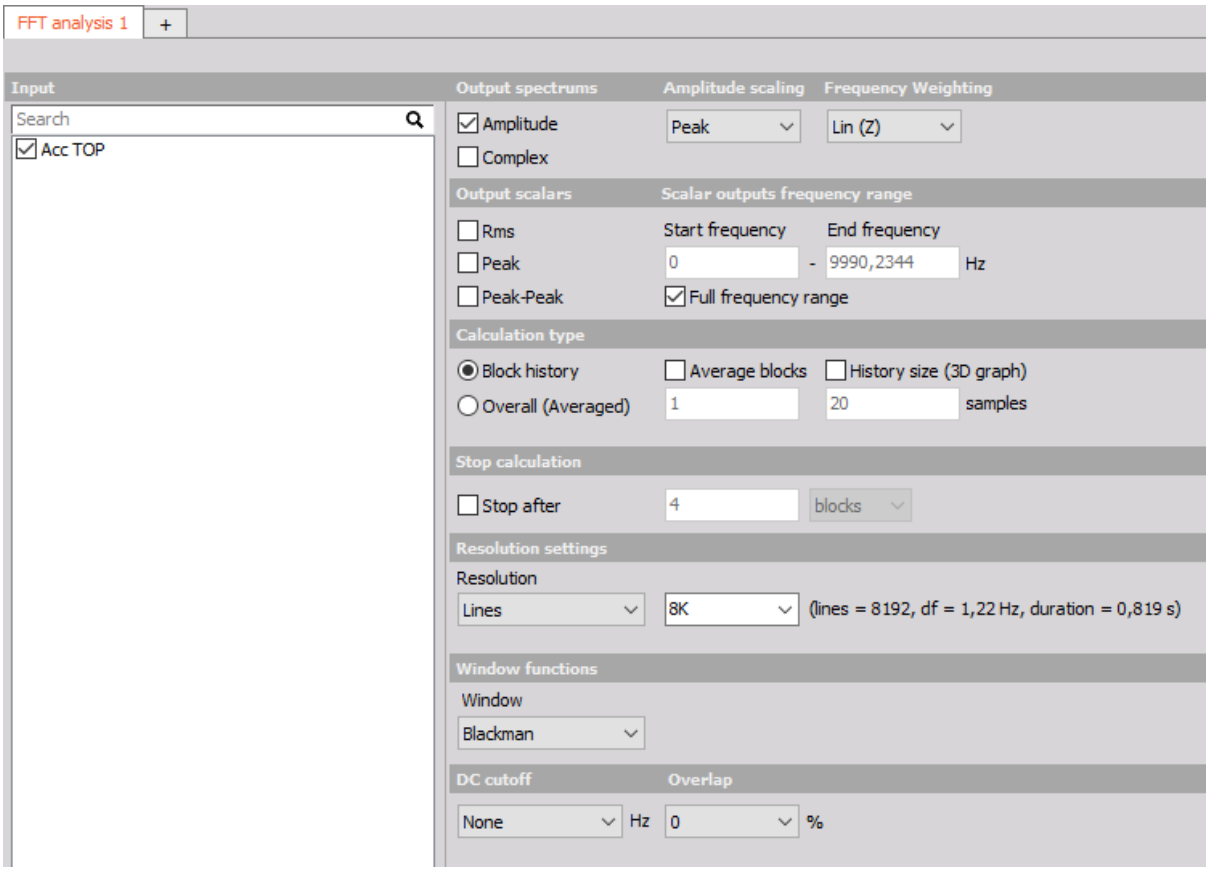


Image 55: FFT analysis setup

Before we run the machine, let's add a visual control 3D graph.

Select the design button and add a 3D graph widget.

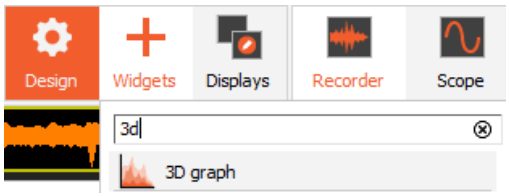


Image 56: Adding a new 3D graph widget

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.

When we run the machine, we clearly see the first harmonic. On 2D graph, we will see amplitude [m/s2] plotted against frequency [Hz].

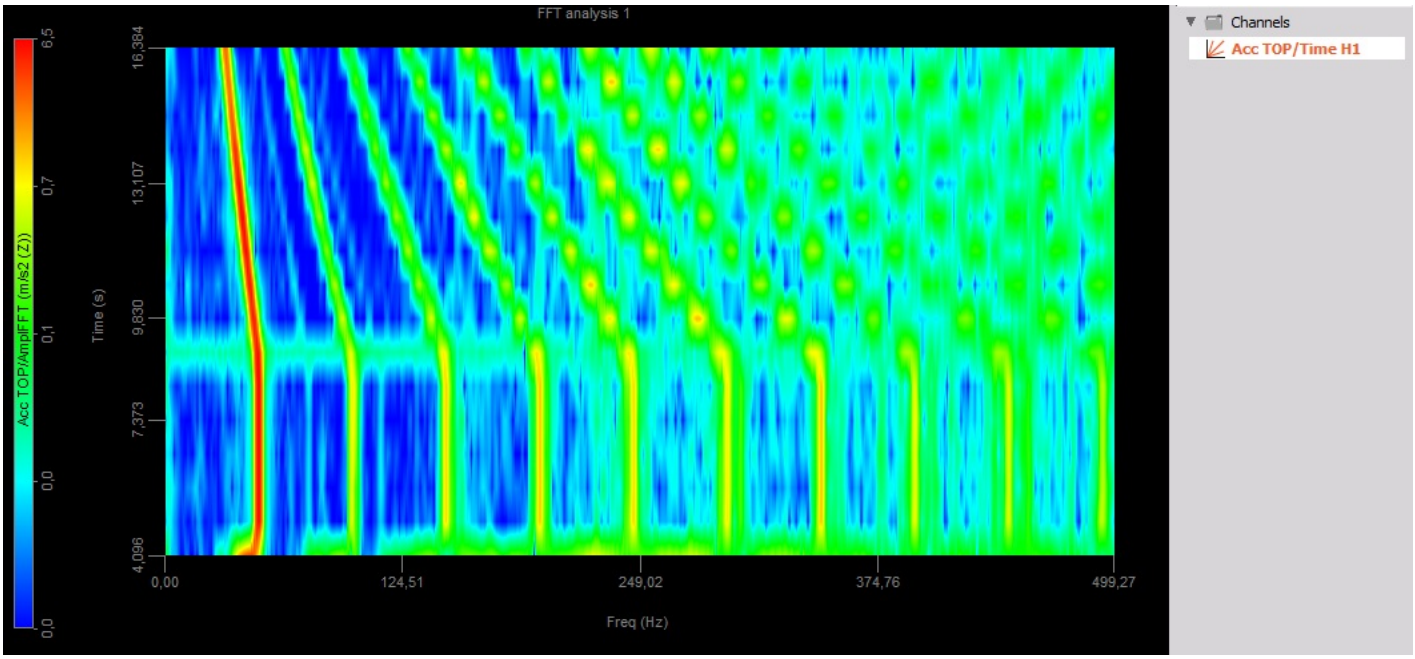


Image 57: FFT results displayed on a 3D graph (FFT vs. time)

If we want to change the view from 2D to 3D we just select different projection types.

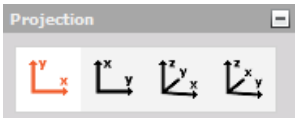


Image 58: Different option from projection on 3D graph

Now we have added also a time domain to our graph - if we take a look at a signal from the FFT analyzer in a 3D graph, we can see how the harmonics are evolving in time.

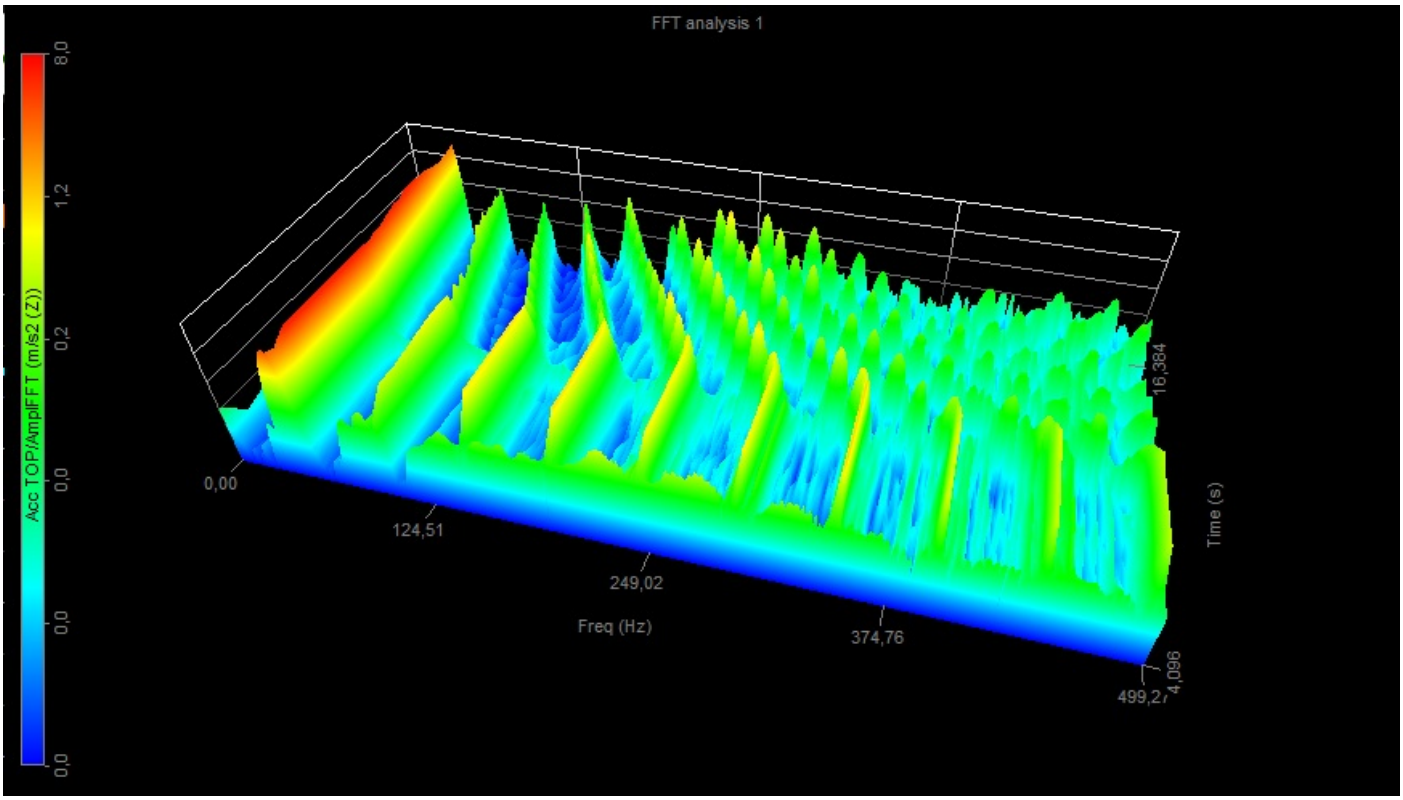


Image 59: Displayed harmonics vs. time on 3D graph

FFT markers