FREQUENCY ANALYSIS
FAST FOURIER TRANSFORM, FREQUENCY SPECTRUM

FOURIER SERIES

In the exercises of module "FUNCTIONS IN MATLAB – STRUCTURED PROGRAMMING" you have developed a function that can generate a cosinusoidal (harmonic) signal {x} with the given amplitude x_o , frequency f in Hz and the initial phase φ_o in degrees for the given time vector {t}, according to the equation $x(t) = x_o \cos(2\pi ft + \varphi_o)$. In this Module you will use this function to generate signals that comprise a known blend of various harmonics to practice their decomposition into the frequency components with the Fourier Transform spectrum.

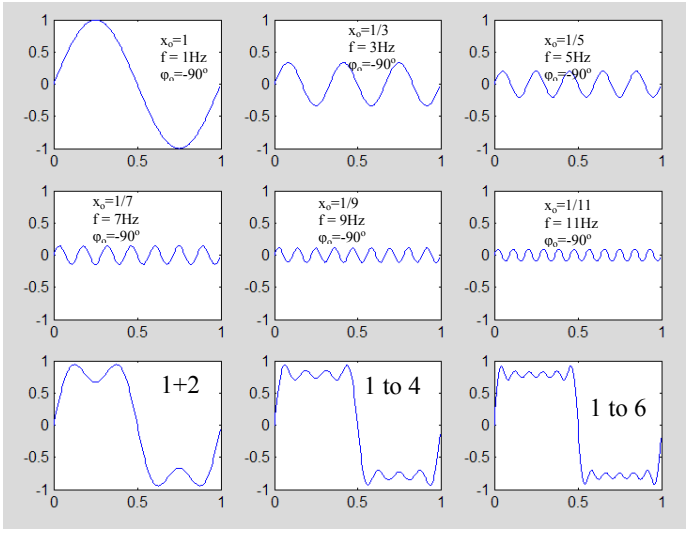


Figure 1 First six frequency components of a square wave

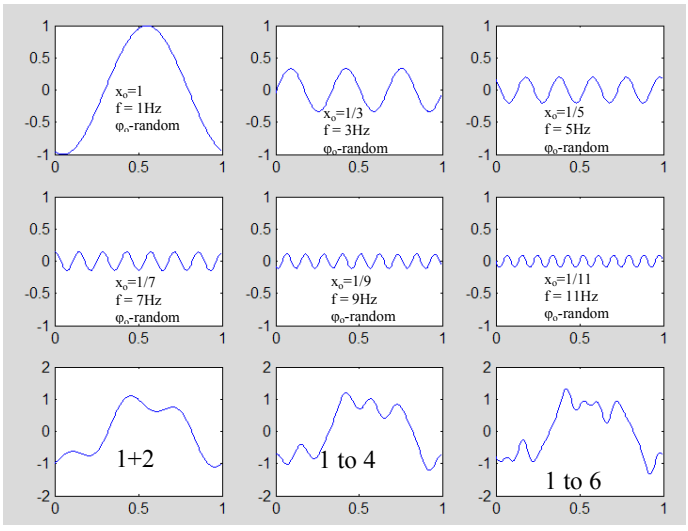


Figure 2 First six frequency components of a random signal

Figure 1 and Figure 2 are illustrations of the Fourier Theorem and the Fourier Series.

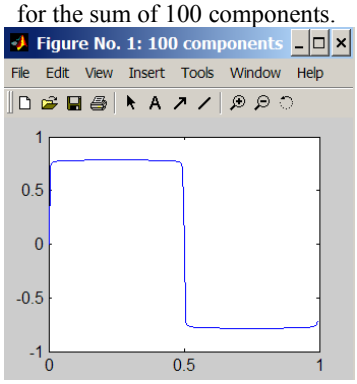
Notice that a sinusoid is the same as cosinusoid with the initial phase of -90°.

The effect of summing more and more terms of a series of sinusoids in the time span of one second by applying the following pattern of amplitude:
1, 1/3, 1/5, 1/7, 1/9 etc and the corresponding pattern of frequency:
1, 3, 5, 7, 9 etc.

The summations of the first 2,4 and 6 terms for the two cases are illustrated are illustrated in Figure 1 and Figure 2.

Two cases are shown:

- Figure 1 - the initial phase is -90° for all cosinusoidal terms, i.e. signals start from 0 and are sinusoids. The combination produces a square wave as seen below for the sum of 100 components.



- Figure 2 - the initial phase of each cosinusoid varies and is allocated from the uniform random distribution between –π and +π radians (i.e. -180° and 180°), i.e. signals start randomly. The combination produces a random signal.

FOURIER SERIES

A periodic function can be represented as a sum of infinite number of (co-)sinusoidal components at equally spaced frequencies with the interval of 1/T, where T is the period of the function, the so called Fourier Series.

FREQUENCY SPECTRUM

When harmonic components of a signal are known, the signal can be presented in a different way that highlights its frequency content rather than its time domain content. Introducing the third axis of frequency perpendicular to the amplitude-time plane the harmonic components can be plotted in the plane that corresponds to their frequencies. For example, a random signal similar to the one in Figure 2 can be presented as shown in Figure 3 in such a way that both time and frequency details are visible.

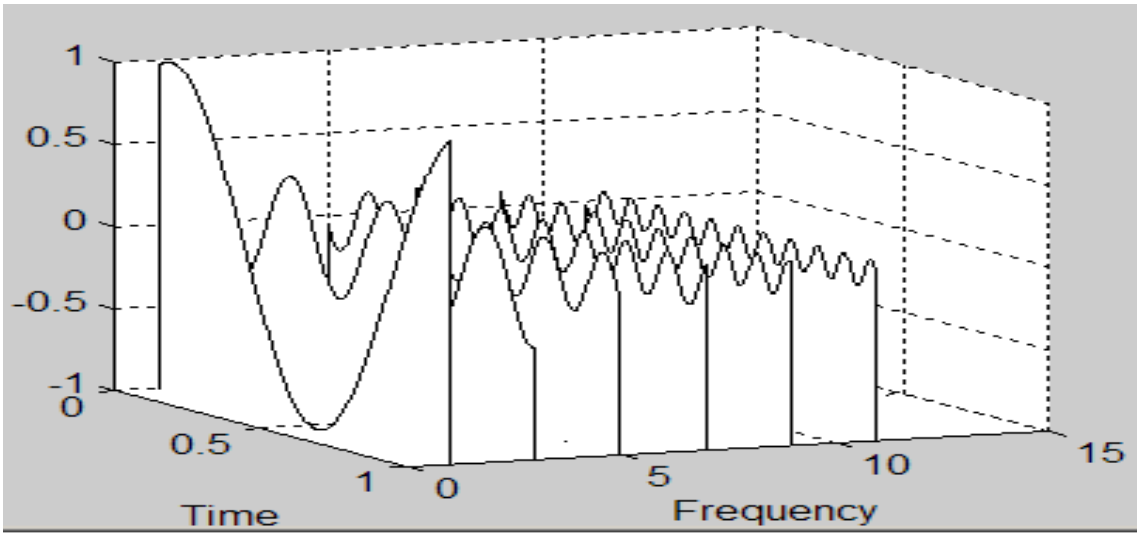


Figure 3 Time-Frequency-Amplitude representation of a random signal

The graph in Figure 3 can be rotated in such a way that the time axis is perpendicular to the observer. This **frequency domain** view when the time axis is no longer visible and only positive values of magnitude of each sine are plotted, is called the **magnitude spectrum**. Figure 4 depicts the magnitude spectrum of the signals shown in Figure 1, Figure 2 or Figure 3.

Components of this spectrum appear as lines to reflect the fact that they are planes in which the cosines are placed. A spectrum however can be plotted showing only the end points of each line connected together. For a spectrum with a large number of points the gaps between the lines are barely visible and the spectrum appears continuous. Irrespectively, the components of a spectrum are traditionally called the lines.

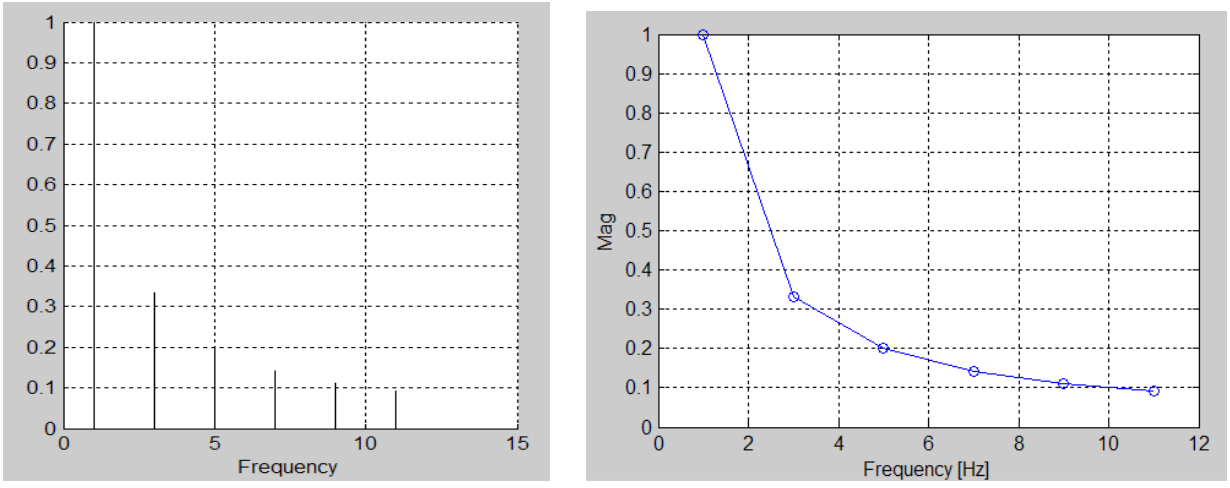


Figure 4 Magnitude spectrum

Magnitude spectrum is not sufficient to fully define the signal. The phase spectrum of each frequency component must be known in order to unambiguously describe it. It is clear now that a component of the spectrum at the given frequency must be described with two parameters. The pair of magnitude and phase is one example of these two parameters.

Example of an Application

An example of an automotive application of frequency spectrum is demonstrated in The roof of a car vibrates excessively at certain driving conditions. The vibrations signal measured with an accelerometer is complex and contaminated, and it is not clear what the cause of this resonance is. However the magnitude spectrum of acceleration reveals a peak at 30 Hz which, after some analysis, coincides with the RPM of one of the shafts in the gearbox which rotates at 1,800rpm at these driving conditions.

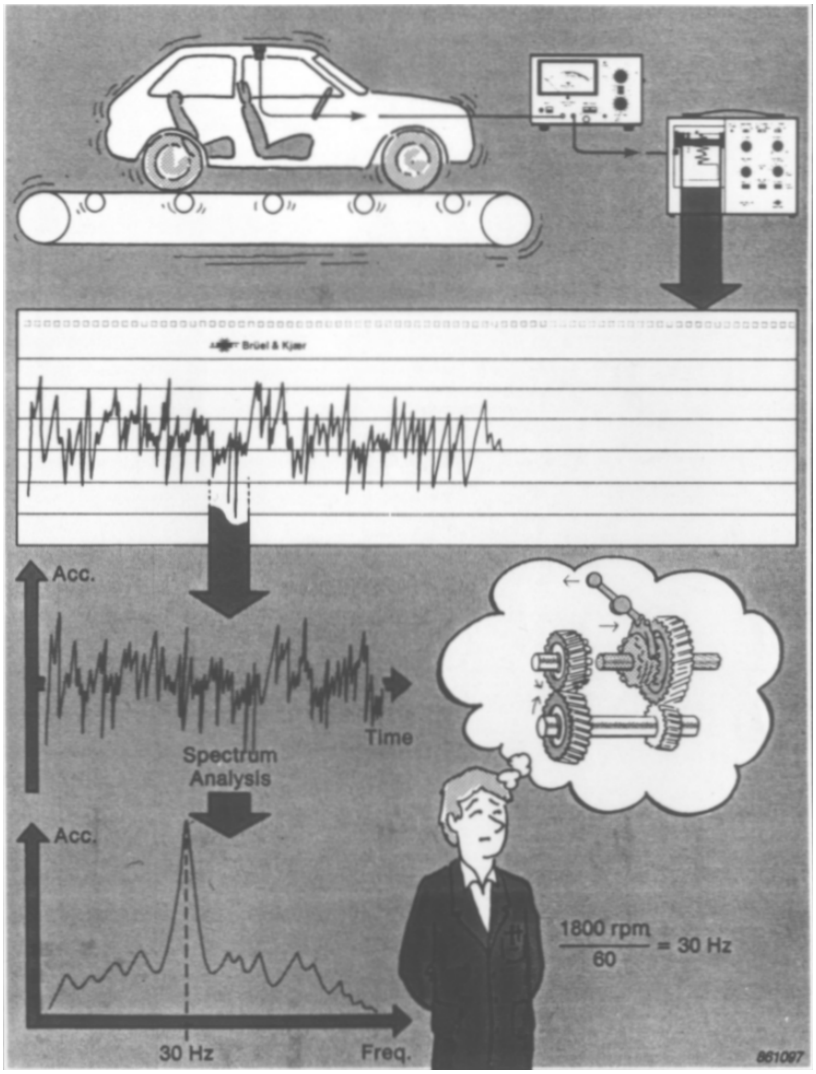


Figure 5 An automotive application of frequency spectrum (Brüel&Kjær "Structural Testing")

FREQUENCY ANALYSIS

In engineering practice signals are usually not described by mathematical functions but come from measurement acquired with a selected sampling interval Δt (its inverse is the sampling frequency or rate $\mathbf{f_s}$). Thus the signal is **not continuous** but **discrete**. They may also be obtained from a simulation models, eg. in Matlab Simulink, which also produce discrete signals. The duration of a signal is finite and in most cases it will not be the same as the period required by the Fourier Theorem. The signal may not be periodic at all.

Due to these limitations:

The purpose of frequency analysis is to devise a method to extract an estimate of frequency components which are not known a priori. The process is known as the Discrete Fourier Transform (DFT)

Awareness of some concepts is necessary in order to understand the DFT or its special case, the Fast Fourier Transform (FFT).

Selection of Sampling Frequency
Shannon Sampling Theorem. Aliasing

The choice of sampling frequency is arbitrary. The signal contains various frequency components to be detected. What are the pitfalls of selecting a "wrong" sampling frequency?

If the frequency to be detected is **f** and the sampling frequency is too low, i.e. $\mathbf{f_s <= 2f}$, a lower erroneous frequency will be perceived, as illustrated in Figure 6. The phenomenon is called the **aliasing**.

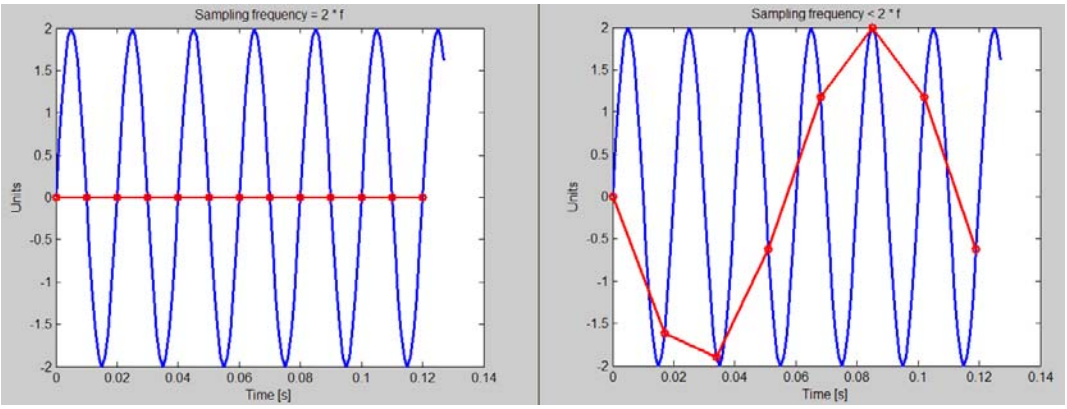


Figure 6 Illustration of the phenomenon of frequency aliasing

Shannon Sampling Theorem is formulated:
The measured signal must not contain frequencies above half of the sampling rate.

Frequency equal to half of the sampling frequency is the highest detectable frequency. It is called the **Nyquist or folding frequency**. The most effective way of preventing the aliasing is by filtering it with a low pass filter with the cut-off frequency less than half of the sampling rate.

Vector Representation of a Cosine

A two-dimensional vector in the complex plane is shown in Figure 7

Complex notation

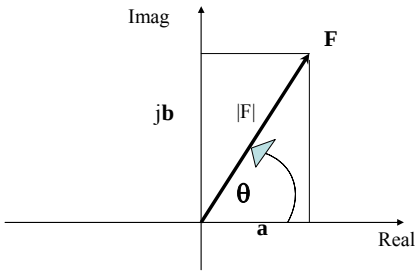
$\mathbf{F = a + j\,b}$

$\mathbf{a} = |\mathbf{F}| \cos\theta$

$\mathbf{b} = |\mathbf{F}| \sin\theta$

$|\mathbf{F}| = \sqrt{a^2 + b^2}$

$\theta = \tan^{-1}\,\mathbf{(b/a)}$



$\mathbf{F = |F|\,(\cos\theta + j \sin\theta)}$

Euler's relationship

$\mathbf{e^{j\theta} = (\cos\theta + j \sin\theta)}$

$\mathbf{e^{-j\theta} = (\cos\theta - j \sin\theta)}$

$\mathbf{F = |F|\,e^{j\theta}}$

Figure 7 A vector in the complex plane with the imaginary axis pointing upwards

Although the complex notation and Euler's expression simplify the mathematics by explicitly removing trigonometric functions, they introduce an unwanted imaginary term.

As shown in Figure 8, on the axes rotated in such a way that the real axis is pointing upwards, the sum of the vector and its complex conjugate (i.e. a vector with imaginary part of opposite sign) is real. If these two vectors are counter-rotating with the same frequency in the complex plane, their sum remains real, change sinusoidally with time and the sum's magnitude is twice bigger compared to the component vector, as demonstrated in . The vectors rotate with a constant frequency but in the opposite directions. The sign of the counter-rotating vector's frequency is opposite to the frequency of other vector. Since the positive angle is measured from the real axis to the imaginary, the anticlockwise rotating vector has a positive frequency. **The frequency of the counter-rotating vector (in the clockwise direction) is negative.**

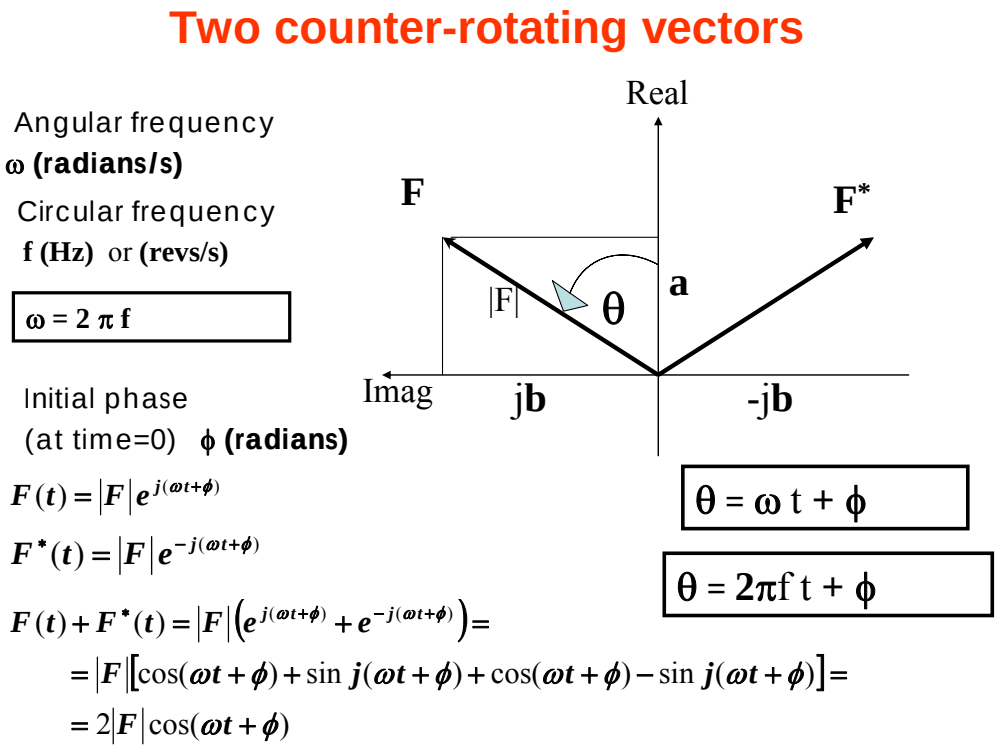


Figure 8 A vector and its complex conjugate rotating in opposite directions (complex plane is rotated - real axis points upwards)

A cosine with the given amplitude and frequency can be represented by two counter-rotating vectors with that frequency with the initial phase $\phi = 0$ and whose magnitudes is equal to half the cosine amplitude. For a sine $\phi = -90^o$

Figure 9 illustrates that concept. An animation can be found at http://staff.vu.edu.au/msek.

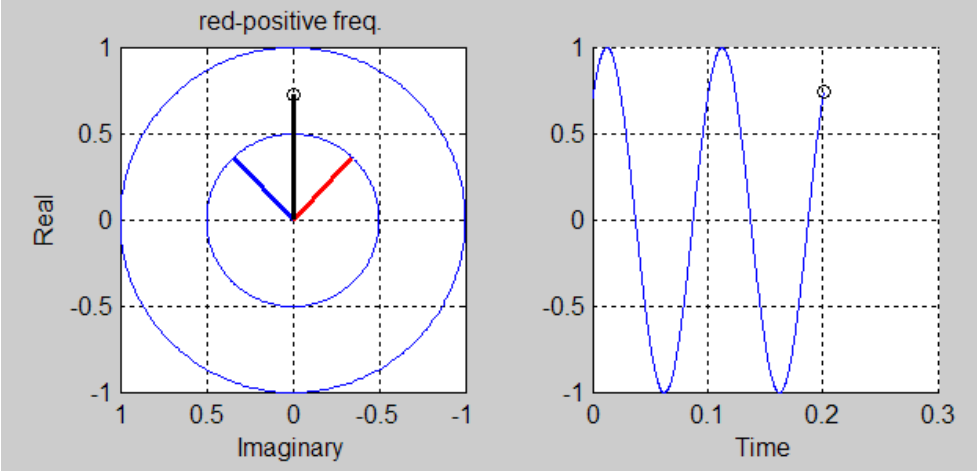


Figure 9 A sum of two counter-rotating vectors drawing a cosinusoid (see http://staff.vu.edu.au/msek for the animation)

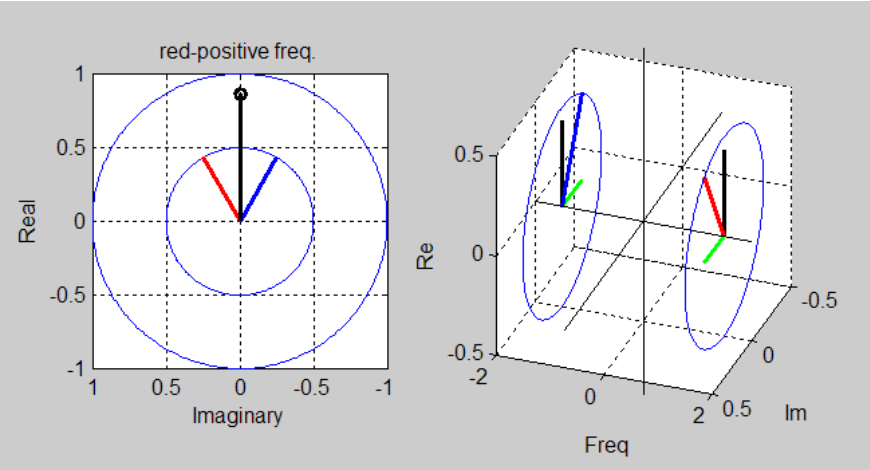


Figure 10 3-D representation of the positive and the negative frequency components of a spectrum (see http://staff.vu.edu.au/msek for the animation)

FOURIER TRANSFORM

As seen, a signal can be thought of as a collection of rotating vectors (or frequency components). In order to extract one of these vectors, one needs to make it stationary, mathematically speaking, for the duration of observation and then integrate over the time of observation. Multiplying a vector $e^{j(\omega t+\phi)}$, which rotates with frequency ω, by $e^{-j\omega t}$ produces a vector with the ωt eliminated, i.e. stationary vector: $e^{j(\omega t+\phi)}e^{-j\omega t} = e^{j\phi}$. Vectors with other frequencies continue to rotate and their integration (or summation) will be close to zero.

It is now clear that the continuous Fourier Transform of signal g(t), which extracts its component G at the frequency ω, is defined as

$$G(\omega) = \int_{-\infty}^{+\infty} g(t)e^{-j\omega t}dt$$

The equation is impractical and for a signal sampled at discrete times t_n and over the finite duration kΔt, the so called Discrete Fourier Transform (DFT) is used

$$G(\omega) \approx \sum_{n=1}^{k} g(t_n)e^{-j\omega t_n}$$

In practice, the circular frequency f in Hz is used instead of ω in rad/s. For such a case

$$G(f) \approx \sum_{n=1}^{k} g(t_n) e^{-j2\pi f t_n}$$

These Fourier Transform equations must be applied to both the positive and the negative frequency.

When f= 0 the sum degenerates to the sum of the values in the signal which is real. In order to produce the mean value, the result needs to be divided by the number of points, k, in the signal. The same scaling needs to be applied to all frequency components of the spectrum in order to produce the correct estimate of the magnitude. At f= 0 the so called DC-value (or the mean value) of the signal is obtained. In many cases in engineering practice the DC value is, or should be 0 even if a measurement does not indicate so, e.g. the mean acceleration of a vibrating and stationary object.

For a signal sampled with the sampling rate f_s , due to the limitation imposed by the Shannon Theorem, components of the spectrum can only be extracted for frequencies between $-\frac{f_s}{2} \le f \le \frac{f_s}{2}$ or between the negative and positive Nyquist frequency. The equi-spaced components are at discrete frequencies and are separated by $\Delta f = \frac{1}{T} = \frac{1}{k\Delta t} = \frac{f_s}{k}$ [Hz]. The Δf is called the frequency or spectral resolution. The +/-Nyquist frequency components of the spectrum are equal the same and are real as their rotating vectors always appear at the same location along the real axis, irrespective of the sign of frequency.

It is important to notice that the Δf depends solely on the duration of observation (or measurement) T, i.e. the longer the duration the better (the smaller) the resolution. For the same number of points k in the sample the resolution can be improved (reduced) by having a worse time resolution Δt or a slower sampling frequency f_s, .

FAST FOURIER TRANSFORM (FFT)

The Fast Fourier Transform (FFT) is an algorithm for calculation of the DFT first published in 1965 by J.W.Cooley and J.W.Tuckey. It has revolutionised the modern experimental mechanics, signal and system analysis, acoustics, and paved the way for the introduction of modal analysis. The FFT algorithm applies only to signals comprising a number of elements which is equal to 2^m (e.g. $2^8 = 256$, $2^{10}=1024$ etc.). Its main advantage is that it significantly reduces the computation time by a factor of the order $m/\log_2 m$, i.e. more than 100 times for a sample of 1024 elements.

From the earlier discussion it is clear that the FFT should return a set of complex numbers, with exception of the spectral components at f=0 and $f=f_s/2$ (the Nyquist frequency), which are both real. The number of FFT elements is equal to the size of the time sample. The second half of these complex numbers corresponds to negative frequencies and contains complex conjugates of the first half for the positive frequencies, and does not carry any new information.

Implementations of FFT algorithm are commonly available:

- In MS Excel® the Fourier Analysis can be found under Tools->Data Analysis.
- In Matlab® there is a function fft()

General Comments on FFT

In order to correctly interpret the results of FFT, one must understand how FFT returns the results of the transform. The following is common to most implementations of FFT including MS Excel® and Matlab®.

Assuming that the number of points k, being a power of 2, was sampled with frequency $f_s = 1/\Delta t$ then

- The frequency resolution $\Delta f = f_s/k$.
- The maximum frequency of the spectrum is the Nyquist frequency = $f_s/2$.
- Since the FFT only does the summation of terms, the values returned by FFT must be scaled by dividing them by the number of points, k.

FFT returns k complex numbers with spectral components as follows:

- The first item is for f=0. Since it has to be real, its imaginary component is 0.
- The next k/2 items are for all positive frequencies in increasing order from Δf , 2Δf, 3Δf,…, (k/2-1)Δf, (k/2)Δf or $f_s/2$. The element for the Nyquist frequency is real and is shared with the one for $f = -f_s/2$, since they are equal.
- The next k/2-1 elements contain the negative frequency components. They are also placed in mathematically increasing order from the most negative frequency f= -(k/2-1)Δf until f = -Δf. In other words, they are sorted from maximum negative to minimum negative frequency. In this sense they appear to be in reverse order of frequency.

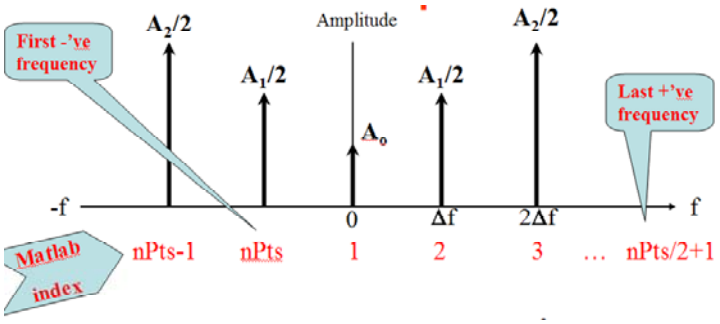


Figure 11 Location of FFT elements in Matlab. In MS Excel the index is represented by the row

Folding FFT Spectrum into a One-Sided spectrum

The negative frequency components are there for mathematical reasons. The spectrum is conjugate even and, as the result, the magnitude spectrum is even, i.e. symmetrical around f=0. Such a spectrum is called two-sided.

For practical reasons we are interested in a presentation of the spectrum for positive frequencies only, the so called one-sided spectrum, similar to Figure 4. However we need to compensate the effect of ignoring the negative frequencies, by multiplying all positive frequency components by 2 (the two counter-rotating vectors are half-size). This process of converting from one-sided to two-sided spectrum is called **the folding**. An algorithm for folding FFT spectrum and calculating and plotting a one-sided magnitude spectrum is depicted in Figure 12.

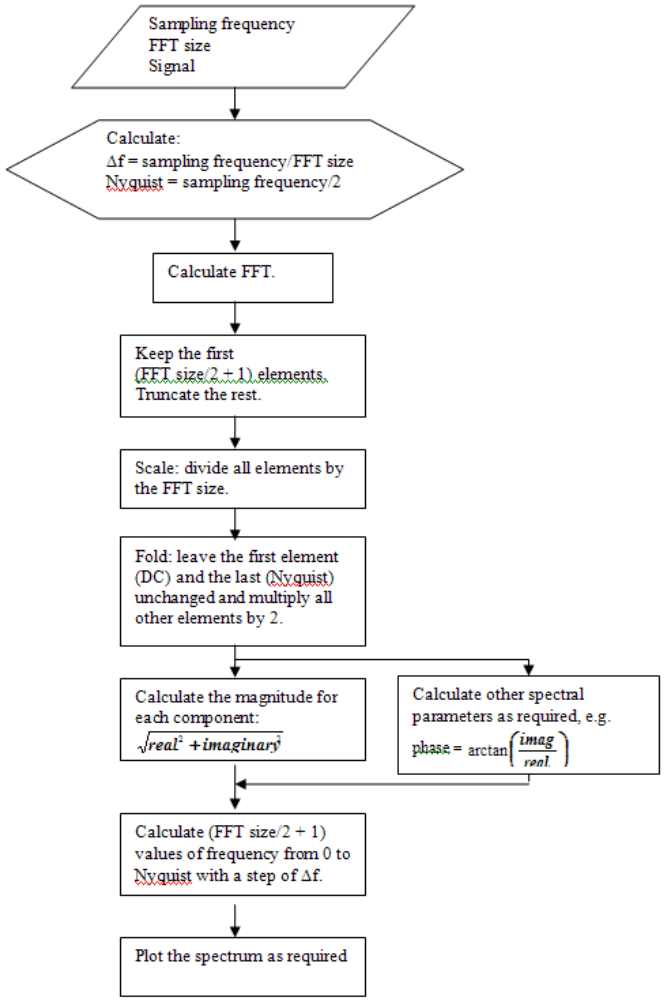


Figure 12 Algorithm for folding the FFT spectrum and producing a magnitude spectrum

IMPLEMENTATION OF THE FFT IN MATLAB

Investigation of the Behaviour of the FFT Algorithm in Matlab

Matlab's function fft() performs the DFT. In order to develop a better understanding of FFT algorithm we will generate a sampled sinusoidal signal with the known amplitude, frequency and phase, and then subject it to FFT and study its behaviour to recover the characteristics of the signal.
A function called 'cosineGenerator' developed as part of the Computations and Engineering Analysis Module "Functions in Matlab – Structured Programming," and shown in the Appendix, will be used.

We choose:

```
amplitude = 2;
frequency = 75;
phase = 0;
samplingFrequency = 750;
numberOfPoints = 2^10;        %ie. 1024 required for FFT
%and generate the signal
cosineGenerator(amplitude, frequency, samplingFrequency,...
    numberOfPoints, phase);
```

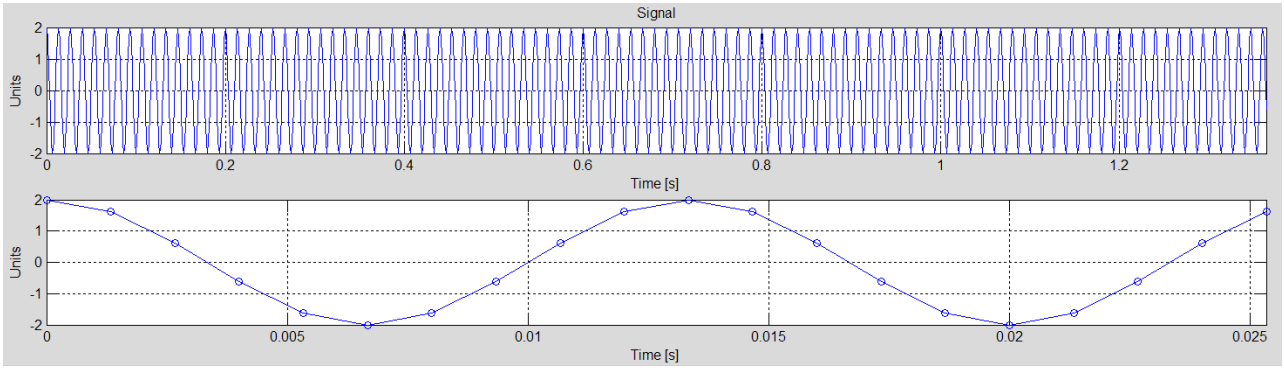


Figure 13 A trial signal and its first 20 points

Let's retrieve the signal, subject it to the FFT and plot the raw, unscaled real and imaginary components:

```
[g,time] = cosineGenerator(amplitude, frequency, samplingFrequency,...
    numberOfPoints, phase);
spectrum = fft(g,numberOfPoints);
figure
subplot(2,1,1)
plot(real(spectrum))
grid on
ylabel('Real')
title('Raw FFT spectrum')
subplot(2,1,2)
plot(imag(spectrum))
grid on
ylabel('Imaginary')
xlabel('Index number')
```

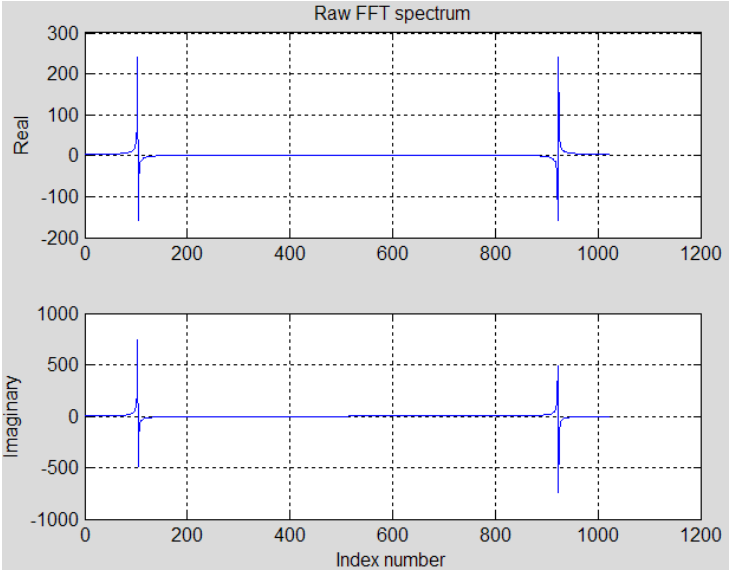


Figure 14 Real and imaginary components of FFT spectrum as returned by fft()

Let's verify that the DC and Nyquist components are real

```
%first two elements of spectrum, DC is the first
DCfirst = spectrum(1:2)
```
>> DCfirst =
  3.6180
  3.6182 + 0.0581i

```
indexNyquist = numberOfPoints/2+1;
%vicinity of Nyquist, which is in the middle
NyquistMiddle = spectrum(indexNyquist-2:indexNyquist+2)
```
NyquistMiddle =
  1.6180 - 0.0123i
  1.6180 - 0.0061i
  1.6180
  1.6180 + 0.0061i
  1.6180 + 0.0123i
>>

It can be seen that the complex numbers surrounding the Nyquist are complex conjugates.

Now let's fold the spectrum according to the algorithm in Figure 12 although some alternative steps will be taken.
The indexNyquist is the same as the number of points in the one-sided spectrum that we want to keep.

```
spectrum = spectrum(1:indexNyquist);      %truncate
spectrum = spectrum/numberOfPoints;       %scale
spectrum(2:end) = 2 * spectrum(2:end);    %compensate for truncating the negative
frequencies
%Calculate the magnitude
magnitude = abs(spectrum);          %abs for complex numbers returns their magnitude

df = samplingFrequency/numberOfPoints;    %frequency resolution
frequencyAxis = [0:indexNyquist-1]'*df;   %values on frequency axis
```

Various Plots

```
figure
plot(frequencyAxis, magnitude)            %plot in a new figure
grid on
xlabel('Frequency [Hz]')
ylabel('Magnitude [units]')

%Position and value of maximum magnitude
```

```
[magMax,indexMax] = max(magnitude);
freqMax = frequencyAxis(indexMax);
%Display in title
title(sprintf('Max magnitude=%f @ %fHz, \\Deltaf=%fHz', magMax, freqMax,df))
%Plot the spectrum as lines using stem function and removing the markers at
%the end of lines
handleStem=stem(frequencyAxis,magnitude);
set(handleStem(1),'Marker','none')
grid on
xlabel('Frequency [Hz]')
ylabel('Magnitude [units]')
%Display in title
title(sprintf('Max magnitude=%f @ %fHz, \\Deltaf=%fHz ',...
    magMax, freqMax,df))
```

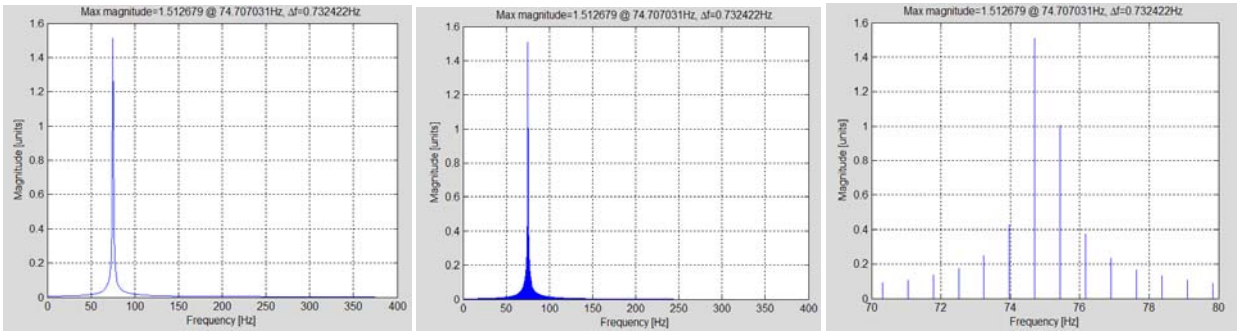


Figure 15 Various plots of the same magnitude spectrum: scattered, stem and stem zoomed around the maximum

Comments

Ideal magnitude spectrum of the analysed signal should be a single line at 75Hz. The frequency of the signal was recovered very well, within the resolution. The magnitude (or amplitude), which should be 2, was obtained with worse accuracy, due to the limitation of DFT such as the leakage (spread) to the adjacent frequencies and the effect of the discrete character and the lack of periodicity in the sample. The means of reducing these adverse effects will not be covered here.

APPENDIX

```
function [s,t] = cosineGenerator(A,f,sFreq,npts,phase)
%   Usage:    [s,t] = cosineGenerator(A,f,sFreq,npts,[phase=0])
%Function returns a time vector t and the corresponding harmonic (cos) wave s
%of length npts with amplitude A and frequency f [Hz],
%sampled at frequency sFreq[Hz or samples per second].
%The argument 'phase' is in degrees and is optional. When the phase is omitted
%it is assumed that the phase is zero. For a sine use phase=-90.
%When no LHS arguments are used (nargout is 0) the sgnal is plotted.

if nargin == 4,
    phase = 0;
elseif nargin < 4 || nargin > 5,
    help cosineGenerator
    error('Incorrect use of the function');
end
%convert phase to radians
phase = phase / 180 * pi;
%sampling interval
dt = 1 / sFreq;
%generate the time vector
t = (0:npts-1)' .* dt;
%generate the wave
s = A .* cos(2 * pi * f .* t + phase);
%plot if no values are to be returned
if nargout == 0
    subplot(2,1,1)
    plot(t,s)
    v=axis;
    v(2)=t(end);
    axis(v)
    grid on
    xlabel('Time [s]')
    ylabel('Units')
    title('Signal')
    subplot(2,1,2)
    pts2plot = 20;
    plot(t(1:pts2plot),s(1:pts2plot),'o-')
    v=axis;
    v(2)=t(pts2plot);
    axis(v)
    grid on
    xlabel('Time [s]')
    ylabel('Units')
end
```