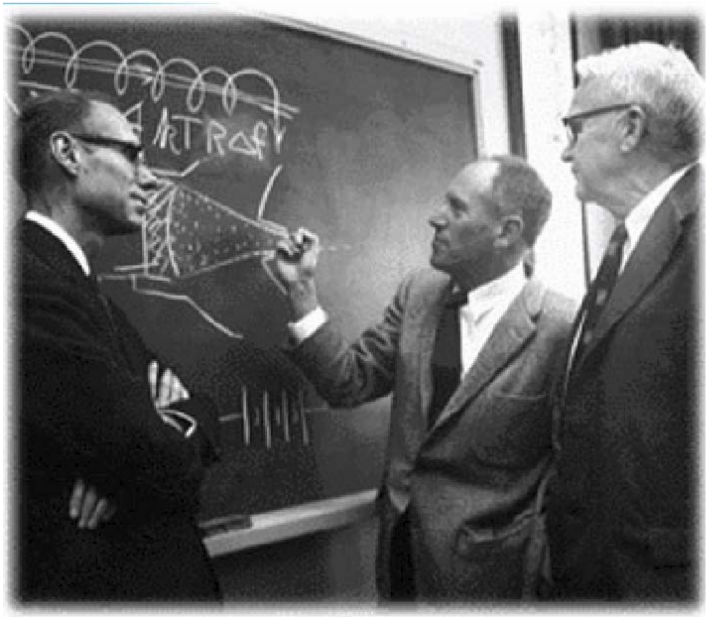
CHAPTER 5

TIME DISCRETE SIGNALS, DFT AND FFT

This chapter discusses how signals are converted into a suitable form to be handled by a computer and how frequency analysis is performed on the signal. First the signal has to be sampled, that is measured at discrete time points and digitised. One of the fundamental rules in handling sampling is now known as the Nyquist-Shannon sampling theorem. In 1927 Swedish born Harry Nyquist determined that an analog signal should be sampled at twice the frequency of its highest-frequency component in order to be converted into an adequate representation of the signal in digital form. The theorem was only formally proved by Claude E. Shannon in 1949.



Harry Nyquist (right) 1944 with colleagues at Bell Laboratories

The development of modern frequency analysers in the 1970's is often attributed to the publication of the FFT (Fast Fourier Transform) algorithm by Cooley and Tukey in 1965. It has however been shown that the algorithm was known by the famous German scientist Carl Friedrich Gauss already 1817. This is even more remarkable as we usually say that the Fourier Transform was introduced by Jean Baptiste Fourier in a book published in the year 1822!

Johann Carl Friedrich Gauss (1777-1855).

5.1 Sampling

When we are handling signals in a digital system, such as a computer, the signals must first be sampled. In practically all cases, the signals are sampled at regular time intervals, Δt. The inverse of Δt is called the **sampling frequency**, f_s. As an example, the **sampling interval** 1 ms corresponds to a sampling frequency of 1 kHz.

$$f_s = \frac{1}{\Delta t} \tag{5-1}$$

What happens to the signal when it is sampled? Can the sequence of samples represent the original continuous signal? Is there an error introduced? We study a signal *x(t)* that is sampled with sampling frequency f_s. The original signal before sampling is continuous in time; see an example in figure 5-1.

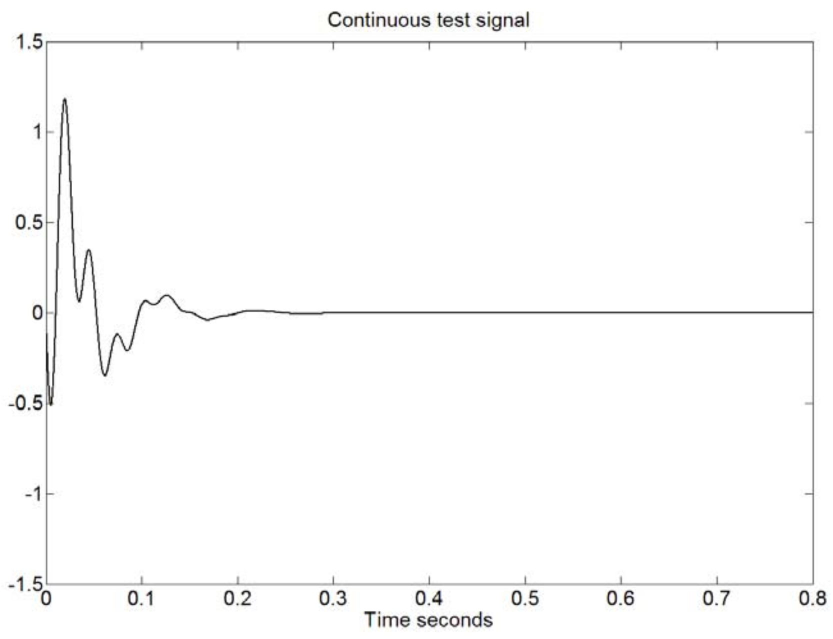


Figure 5-1 Test signal.

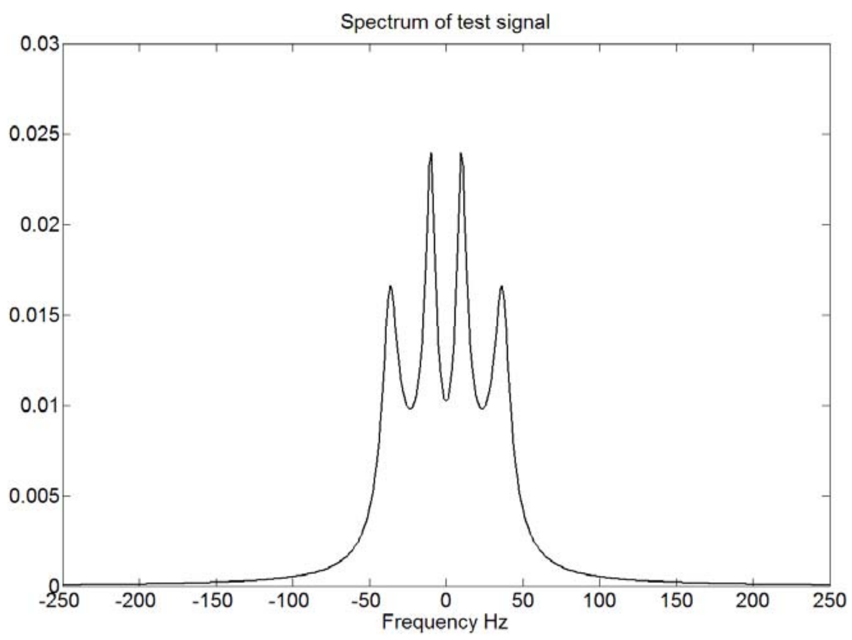


Figure 5-2 Fourier spectrum of the test signal

The Fourier transform of the test signal, the signal spectrum, is given in figure 5-2. We present it for both positive and negative frequencies, for reasons explained below. We can get a basic result by assuming that a sampled signal, *y(t)*, is obtained by multiplying the original *x(t)* with a sampling signal, *s(t)*, made up of a train of short pulses with a pulse frequency of f_s, see figure 5-3.

$$y(t) = x(t) \cdot s(t) \tag{5-2}$$

A part of the result after the multiplication, *y(t)*, is given in figure 5-4.
As *s(t)* is periodic, we may write it as a complex Fourier series

$$s(t) = \sum_{m=-\infty}^{\infty} c_m \, exp(-im\omega t) \; , \tag{5-3}$$

leading to

$$y(t) = \sum_{m=-\infty}^{\infty} c_m \, exp(-im2\pi f_s t) \cdot x(t) \; . \tag{5-4}$$

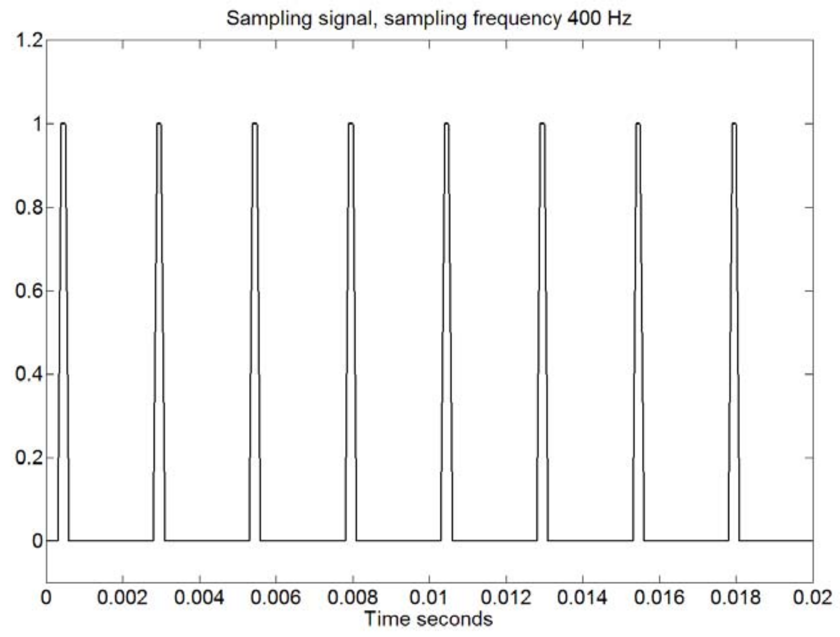


Figure 5-3 Sampling signal, a pulse train used to sample the test signal.

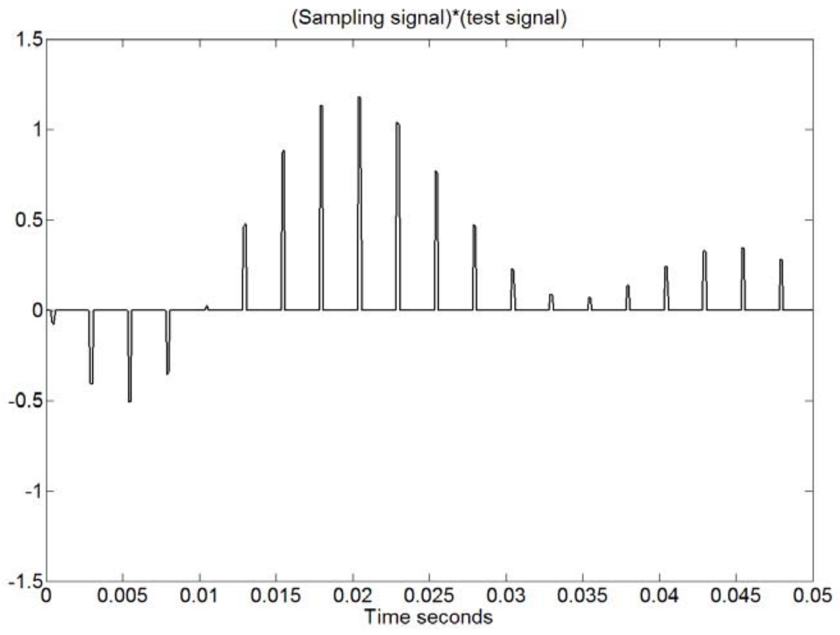


Figure 5-4 The sampled test signal, a multiplication of the test signal with the pulse train in 5-3.

We now take the Fourier transform of *y(t)*, using equation (3-29)

$$F\{x(t)\cdot\exp(i2\pi f_0 t)\} = F_x(f - f_0)\,, \tag{5-5}$$

which gives

$$F\{y(t)\} = F\left\{\sum_{m=-\infty}^{\infty} c_m \exp\left(-im2\pi f_s t\right)\cdot x(t)\right\} = \sum_{m=-\infty}^{\infty} c_m \cdot F_x\left(f - mf_s\right). \quad (5\text{-}6)$$

The result is shown in figure 5-5.

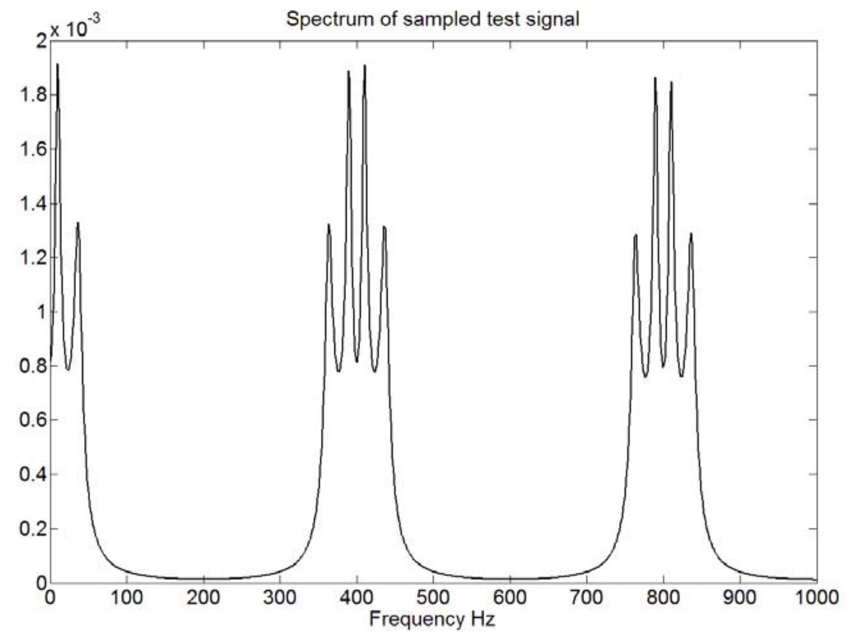


Figure 5-5 Fourier spectrum of the sampled test signal.

We find that sampling a signal with sampling frequency f_s leads to a periodic repetition of the signal spectrum in the frequency domain. The original spectrum will be repeated, centred on integer multiples of the sampling frequency. (If the sampling pulses are made shorter and shorter, the coefficients c_m will end up constant). We also see that thinking of the original spectrum as defined for both positive and negative frequencies makes the understanding easier.

This very fundamental finding will have great impact on the way we handle sampled signals in digital signal processing. In figure 5-5 we find that as long as the original signal spectrum has no contribution for frequencies above half the sampling frequency (200 Hz in the figure), we have an exact duplicate of the original spectrum below half the sampling frequency. The spectrum is only multiplied with the factor c_0; the shape is not distorted. We may then in theory perform an inverse Fourier Transform on that part of the spectrum and get the original signal back. This leads to the following important theorem:

If a signal has a spectrum that has no contribution for frequencies above W Hz, then all information in the signal is retained if it is sampled with a sampling frequency f_s>2W.

This is the **Sampling Theorem**, formulated by Shannon 1948. It means that if we are only interested in the behaviour of a signal up to a certain frequency limit (which is very common) we have no error at all if we sample the signal with a sampling frequency that is at least double that frequency. This is the reason why we may perfectly recognise the voice of a friend talking to us over a telephone connection, where all the signal processing is performed on sampled signals.

We now realise that if we lower the sampling frequency in our example enough, the left part of the spectrum around f_s will interfere with the right part of the original spectrum. This is called **aliasing**. Once the signal is aliased, there is no way to sort out the original signal from the alias. We also see, that a signal frequency just greater than $f_s/2$ gives an aliased frequency that is just below $f_s/2$. It seems as if the frequencies above half the sampling frequency are folded back around $f_s/2$.

To protect us from aliasing, we therefore always use an **anti-aliasing filter** to limit the signal frequency content before sampling. Theoretically, a cut-off frequency of half the sampling frequency would suffice, but in practise, we must choose a somewhat lower cut-off frequency. Usually we reason the other way round. We select a frequency range of interest, say f_{max} = 100 Hz. We then apply an anti-aliasing low pass filter, which is flat up to 100 Hz and then rolls off. We then select a sampling frequency, which is slightly greater than two times f_{max}. For reasons to be explained later, a common choice is 2.56 times f_{max}, in our example this means f_s = 256 Hz.

In figure 5-6 the basic principle for an anti-aliasing filter is shown. The filter has a flat pass band up to f_{max}, where it starts to roll off. Frequencies between f_{max} and $f_s/2$ are ignored. Frequencies between $f_s/2$ and f_s-f_{max} are folded back to the region f_{max} to $f_s/2$ and are also ignored. The first critical frequency is f_s-f_{max}, as signal frequencies from now on will be folded back into the used frequency range. But those signal frequencies are damped in the filter. We want a large dynamic range, which means that we want to be sure that signals with small amplitudes that we find in our used range are genuine. To ensure that we must select an anti-aliasing filter with high damping from f_s-f_{max} and higher.

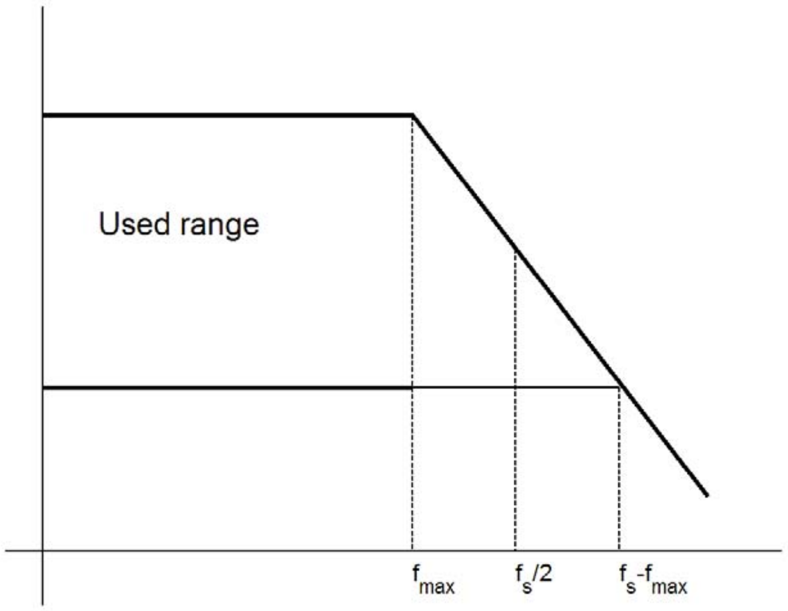


Figure 5-6 Frequency response for typical anti-aliasing filter. The used range is from frequency zero to f_{max}. The lowest frequency folded back to the used range is f_s-f_{max}. From that frequency and up, the damping of the filter must be sufficient.

5.2 Analogue – to - Digital Conversion

The next step in the process of getting a signal into a form that can be used in a computer is to measure the samples. This is done in an **Analogue-to-Digital Converter**, an ADC. If sampling means a discretisation in time, an AD conversion means a discretisation in amplitude, a quantisation. That is because the converter has a limited resolution; it has a limited number of bits. Most ADCs produces a binary output. The input voltage range, for example $\pm$ 10 volts, is divided into 2^n discrete steps; where n is the number of bits. If we have 12 bits, the $\pm$ 10 volts are divided into 2^{12} = 4096 steps, which mean that each step is 20/4096 volts, roughly 5 mV. Wherever the signal sample amplitude is within a 5 mV interval, the binary number corresponding to that interval is given as the ADC output. In figure 5-7 the principle for **quantisation** is shown.

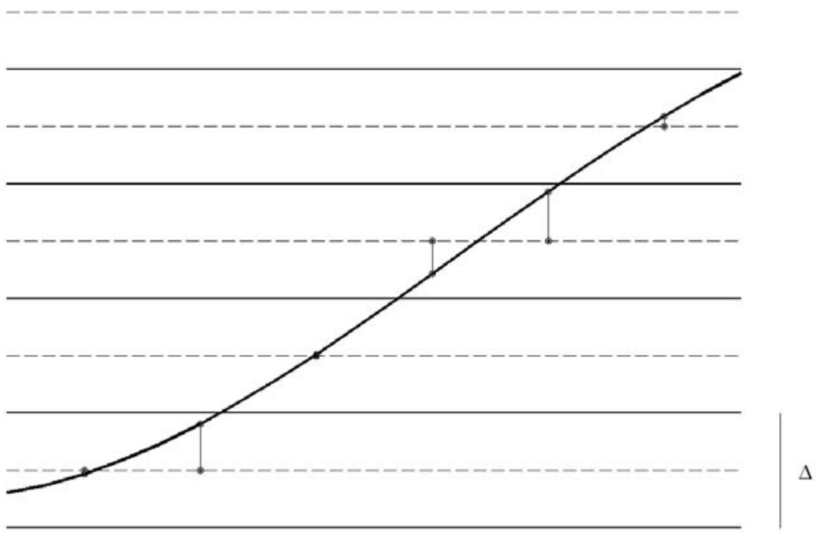


Figure 5-7 The quantisation process.

Each sample is converted into a value that corresponds to the middle of the amplitude interval. A quantisation error is introduced. A commonly used model for this error is to assume that each error is independent of the rest, and that the error amplitude is evenly distributed in the range –Δ/2 to Δ/2, where Δ is the step in the ADC process. We may then calculate the mean square value of the error.

$$\sigma^2 = \int_{-\infty}^{\infty} p(x)\cdot x^2 dx = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \frac{1}{\Delta}\cdot x^2 dx = \frac{\Delta^2}{12} \quad . \qquad (5\text{-}7)$$

By **dynamic range** for a system we mean the ratio between the largest and the smallest signal the system can handle. In the case of AD-conversion it is common to define the dynamic range as the ratio between the largest sine wave the ADC can handle and the quantisation noise. We measure both with their r.m.s.-value. If the ADC has an input voltage range of $\pm\ V$ volts and n bits, the dynamic range will be:

$$\frac{\textbf{\textit{rms of largest sine}}}{\textbf{\textit{rms of quantisation noise}}} = \frac{\frac{V}{\sqrt{2}}}{\frac{\Delta}{\sqrt{12}}} = \sqrt{6} \cdot \frac{V}{\frac{2V}{2^n}} = \sqrt{1.5} \cdot 2^n \quad . \qquad (5\text{-}8)$$

The dynamic range is usually given in dB, so we take

$$\textbf{\textit{dynamic range}} = 20 \cdot {}^{10}log(\sqrt{1.5} \cdot 2^n) = 1.8 + 6n\ dB\,. \qquad (5\text{-}9)$$

A modern ADC has 12 to 16 bits. An instrumentation tape recorder, using the DAT format, usually has 14 bits. This gives us a dynamic range from approximately 72 dB for 12 bit converter to 96 dB for the 16 bit converter. This could be compared to the dynamic range of a high quality FM tape recorder, which typically was 48 to 52 dB.

5.3 The Discrete Fourier Transform, DFT

We have seen in section 5-1 that the sampling of a time signal leads to a periodicity in the frequency domain when we perform a Fourier Transform. The transform in that case was still a continuous function in frequency.

We now take the procedure another step forward by introducing the **Discrete Fourier Transform, DFT**. One way of looking at it is as a sampled version of the complex Fourier series in (3-17), slightly rewritten as

$$x(t) = \sum_{k=-\infty}^{\infty} c_k \, exp(ik2\pi f_0 t)$$

$$with$$

$$c_k = \frac{1}{T}\int_0^T x(t) \cdot exp(-ik2\pi f_0 t)\,dt \quad . \qquad (5\text{-}10)$$

Instead of the continuous function *x(t)* we now have samples of *x*, taken at time intervals *Δt*, the inverse of the sampling frequency f_s. We introduce the notation *x(n)* for the n:th sample of *x(t)*, i.e. *x(nΔt)*. We now look at *N* samples of *x*, spanning a time interval of *NΔt = T* seconds. If this had been one period of a periodic function, the fundamental frequency in the Fourier series would have been $f_0 = 1/T = 1/N\Delta t$. We now calculate the Fourier coefficient c_k as a sum over the samples *x(n)* instead of the integral in (5-10).

$$\begin{aligned} c_k &= \frac{1}{T}\int_0^T x(t) \cdot exp(-ik2\pi f_0 t)\,dt = \\ &= \frac{1}{T}\sum_{n=0}^{N-1} x(n) \cdot exp(-ik2\pi \frac{1}{N\Delta t} \cdot n\Delta t) \cdot \Delta t = \\ &= \frac{1}{N}\sum_{n=0}^{N-1} x(n) \cdot e^{-i2\pi\frac{nk}{N}} \end{aligned} \qquad (5\text{-}11)$$

The last part is the definition of the Discrete Fourier Transform, DFT.

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) \cdot e^{-\frac{i2\pi nk}{N}} \qquad (5\text{-}12)$$

X(k) is a complex function of *k*, which corresponds to the frequency. For *k* = 0 we get the mean value of *x*. For *k* = 1 we find the Fourier coefficient corresponding to one period over the time interval *T*, or the frequency *1/T*. For *k* = 2 we have two periods over *T*, corresponding to a frequency of *2/T*. The step in frequency, *Δf*, is therefore *1/T* or *1/NΔt* or f_s/N.

We find that the transform is periodic with a period equal to the sampling frequency f_s, because if we let $k = k + N$, the sum stays the same. This suits us fine, as we get the same result with the continuous Fourier transform. So, we have N samples of $X(k)$, spanning the interval from zero to f_s (or rather $f_s - \Delta f$) and then it repeats. Corresponding to the DFT from time to frequency, there is an **inverse transform** from frequency to time:

$$x(n) = \sum_{k=0}^{N-1} X(k) \cdot e^{\frac{i2\pi nk}{N}} \tag{5-13}$$

We note that we have a scaling constant $1/N$ when we go from time to frequency and a constant 1 when we go from frequency to time. It is quite common to use scaling constants the other way round, with 1 from time to frequency and 1/N from frequency to time. As we have seen, our scaling is consistent with the Fourier series. As we will find later, the other scaling convention is consistent with the z-transform. **If you use a DFT, find out the scaling, it is important when you scale the spectra you compute!**

5.4 Periodicity in time and frequency

We have noticed that already the sampling gives us a frequency spectrum that is periodic with the period equal to the sampling period. We also noticed that the DFT was periodic with the same period. We introduced the DFT from the complex Fourier series. We could as well have taken it as a sampled (in frequency) version of the continuous Fourier transform of our sampled (in time) signal in a time interval 0 –*T*. If we sample a continuous function in the frequency domain, we would then expect a periodicity in the time domain. The period would then be the inverse of the sampling interval in the frequency domain, in consistency with the period we got when we sampled a time signal. We have $\Delta f = f_s /N$, which means that the period in time would be N/f_s or $N\Delta t = T$. This means that if we study a piece of a continuous time signal in the time interval 0 – *T*, the DFT "thinks" that this is one period of a signal, periodic with the period *T*. This will lead to considerations further on.

So, regardless of how we introduce the DFT, we will arrive at a perfect symmetrical relation; we study signals in the time domain that are sampled and periodic. They correspond in the frequency domain to functions that are sampled and periodic. We summarise the relations between the parameters involved:

Sampling time interval	Δt
Sampling frequency	$f_s = 1/\Delta t$
Number of samples	N
Period in time	$T = N\Delta t = N/f_s$
Frequency resolution	$\Delta f = 1/T = f_s/N$
Period in frequency	$N\Delta f = f_s$

5.5 DFT symmetry properties

We have noticed that our frequency functions computed with DFT on sampled signals are periodic with the period f_s. We may then as well cut out the frequency interval $-f_s/2 - f_s/2$ in the frequency region instead of the “definition interval” 0 - f_s. It is interesting to look for the relationship between the frequency content for positive and negative frequencies. In most cases, our time signals are considered to be real. We then find the following relations between the frequency content *X(k)* for real time signals:

Re{X(-k)} = Re{X(k)}	The real part is an even function in k
Im{X(-k)} = -Im{X(k)}	The imaginary part is an odd function in k
/X(-k)/ = /X(k)/	The magnitude is an even function in k
phase{X(-k)} = -phase{X(k)}	The phase is an odd function in k

Usually we then only show the magnitude of the spectrum, and we realise that we may present it for positive frequencies only, i.e. for $0 < f < fs/2$. (If we have used a proper anti-aliasing filter, we will only show the frequency interval $0 < f < f_{max}$).

5.6 FFT, Fast Fourier Transform

The DFT is a demanding algorithm. To perform an N point DFT, N^2 operations are required, where one operation is defined as one complex multiplication and one addition. (You have to perform N steps in n for N different k). There is a lot of computation time to save by using the symmetry of the trigonometric functions involved in the computation. If you split your N samples of the time signal in two parts, each $N/2$ samples long, you may compute the DFT for each part. That will take $2*(N/2)^2 = N^2/2$ operations. You may then compute the DFT for the whole signal from a combination of the two DFTs that will take N operations, which is small compared to $N^2/2$. (If N is 1024, $N^2/2$ is 512 times greater than N). So, we gain a lot of computations. Then of course, we may compute the DFTs of length $N/2$ by splitting them into two parts and so on. This splitting in two parts show why our algorithms are most efficient when N is a power of 2.

When N is a power of 2, the number of calculations needed will be $N*{}^2log(N)$ instead of N^2. With $N = 1024$, $N^2 = 1048576$ and $N*{}^2log(N) = 10240$. We gain a factor 1048576/10240 = 102.4. So, the computation will take less tan 1% of the original time if we use this smart algorithm. The algorithm is called Fast Fourier Transform, FFT. It exists in many versions. So remember, FFT is just a smart and efficient way to compute DFT!