Introduction to signal processing: sampled signals

E. Tikhonov and M. Sneps-Sneppe

***Abstract* — The article introduces the theoretical foundations of the modern method of storing, transmitting and processing signals: digital processing. It's a way to use relatively small number of values instead of continuous real signal.**

For example, when transmitting information in telephony, speech is concentrated in the bandwidth up to 4 kHz. But we need doubled frequency 8 kHz sampling in digital form. This is a fundamental requirement.

The reason is that the sampled signal is not just a set of quantities, but a set of short scaled pulses of large value. Resulting spectrum has multiple copies of original one infinitely duplicated on sampling frequency steps. If these copies do not overlap significant (due to the limited original useful spectrum) we could restore initial signal with filtering just one of them. That's why doubled sampling frequency is needed.

If the number of samples used is finite (as in the real cases), then we can use only an equal number of samples of the spectrum (or even a half of them). It's enough for restoring.

As in the rest parts of the series, all theoretical ideas are illustrated by mathematical rationales and Matlab programs that demonstrate the work "in life".

***Keywords* — Data rate, Dirac comb, Discrete Fourier Transform (DFT), discrete-time signal, Fast Fourier Transform (FFT), low pass filter, MatLab, Nyquist–Shannon–Kotelnikov theorem, RC-circuit, Sampling, sampling theorem, sinc-filter, telephone communication.**

I. INTRODUCTION

The article continues the introduction to a signal processing tasks for radio astronomy measurements and satellite data collection that could be used in the "Ventspils International Radio Astronomy Center" of the Ventspils University of Applied Sciences.

In previous papers complex signal representation [1] and spectral analysis principles [2] were discussed. Current article is dedicated to basics of discrete approximations of a signal for digital processing.

The rest of the paper is the following. In Section 2 example of digital voice transmit is given (comparing with analogous). Section 3 suggests mathematical description of sampled signal and corresponding spectrum changes. In Section 4 restoring original continuous signal from these samples and in Section 5 restoring the samples from spectrum samples are discussed.

All the Matlab codes (used as the examples and produced illustration figures to all parts) are collected in the Appendix.

Manuscript received May 31, 2019.

Eugene Tikhonov is with Ventspils University of Applied Sciences, 101a Inženieru Street, LV-3601, Ventspils, Latvia (e-mail: abava@abava.net).

Manfred Sneps-Sneppe is with Ventspils University of Applied Sciences, 101a Inženieru Street, LV-3601, Ventspils, Latvia (e-mail: amanfredss@venta.lv).

II. DIGITAL VOICE TRANSMISSION

The basic concept of telephone communication is the so-called channel of tone frequency. Why? Formants of voice that determining speech intelligibility are located mainly in the frequency range from 300 to 3 400 Hz. So for the analog voice telephony, the band frequency 4 kHz is usually allocated for transmission of each channel [6].

Currently, signal processing is usually done digitally - on digital computers (or specialized digital processors). They do spectral analysis, filtering, storage and transmission in digital form [3]. That means only a certain set of signal values are selected from the real signal - *samples*.

If we use the digital transmission, then according to the Kotelnikov theorem one should take the sampling rate of 8 000 counts per second (8 kHz) - twice as much. This is a deep and direct consequence of the discrete nature of the signal obtained from the continuous.

The theorem was proposed and proved by Vladimir Kotelnikov in 1933 in his work "On the capacity of ether and wire in telecommunications", in which the theorem was formulated as follows:

"Any function $f(t)$ consisting of frequencies from 0 to f_C can be continuously transmitted with any accuracy using numbers following each other in $\frac{1}{2f_C}$" [4].

The theorem was discovered by several authors independently. It is thus also known by the names *Nyquist–Shannon–Kotelnikov*, and *cardinal theorem of interpolation*, also sampling theorem, and *Whittaker–Shannon interpolation formula* [5].

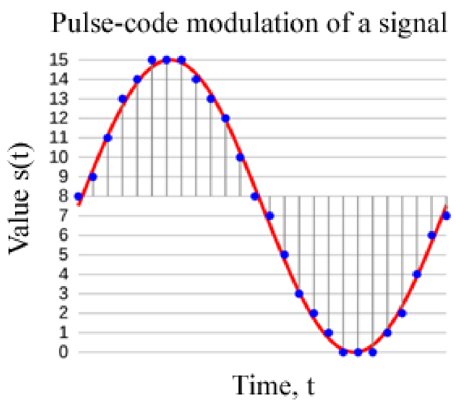


Fig 1. Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals. Sampling and quantization of a signal (red) for 4-bit linear pulse-code modulation (LPCM).

In previous articles of the series [1] and [2], we used MatLab scripts that simulates continuous signals by a large number of points. In fact, we intuitively believe that if we take a lot of samples, we will get the same (or almost the same) result as for a real continuous signal. Is this true, and how many samples should we get?

III. DISCRETE SIGNALS

In digital signal processing (for example for digital computer processing with quantized time ticks) we have a deal with not continuous in time analog signal $s(t)$, but with *discrete signal* (or *discrete-time signal*) – set or sequence of signal value samples set in different times $\{s_n\}$[6]. In most useful cases each value set at equal time intervals T (sampling time):

$$s_n = s(T \times n)$$

Also data values of such signal could be not continuous but also discrete (they can only take on one of a finite number of values) – this type of signal is called *discrete-value signal*. Discrete-time discrete-value signal is called *digital signal*, and it's possible values called *symbols*.

For the important case, where signal has only 2 possible values 0 ("false") and 1 ("true") it is called *bitstream*. Transmitted number in this case represented as a sequence of sequential *bits* $x_i = [0, 1]$:

$$x = \sum_k x_k \times 2^k$$

If signal takes 2^N possible values also say that it contains N bits for each symbol (because each symbol could be represented as N bits). Bitstream obviously is a sequence of 1-bit symbols "0" and "1".

Further we will not pay special attention to the quantification of values, only the discretization of the signal in time.

Parameter of a discrete signal that shows how many values in second contained in a digital signal called signal *data rate*. Signal *bitrate* is a number of bits per second in digital signal (for the case of bitstream is equals to data rate obviously). For example, if $\{s_n\}$ has K symbols per second and each symbol contain M bits, then: data rate is K, bitrate is $K \times M$.

for the voice telephony example discussed earlier, the numerical value of each sample is represented as a 7 or 8-bit binary code (that means each sample value if from 0 to 127 or to 255). So in the world of telecommunications there are two transmission systems - at a speed of 56 (8 kHz * 7 bits) or 64 (8 kHz* 8 bits) kbit/s.

This is not about the amount of useful information contained in the signal in general. An overview of the various encoding and compression methods is not included in this article devoted only to the transmission of the physical values of a given signal.

A. *What is a discrete signal*

The discrete signal fundamentally differs from continuous. It is shown in mathematics that rational numbers (that is, those that can be numbered in principle, even if there are an infinite number) are fundamentally smaller than the real numbers (which belong to the values of continuous signals) that form the "continuum" (continuum) [7]. That is, even the smallest continuous variable signal will be "more powerful" than any number of fixed samples, no matter how many there are.

Due to the fact that the samples are just a set of points (or numbers), we cannot simply use them in the real world instead of continuous signals because of their finite value and zero duration.

Therefore, a discrete signal means a special function that depends on these number samples, used as the height of the corresponding delta functions, following each other after a *sampling time interval* T:

$$s_{sampled}(t) = \sum_{n=-\infty}^{\infty} s_n \delta(t - nT)$$

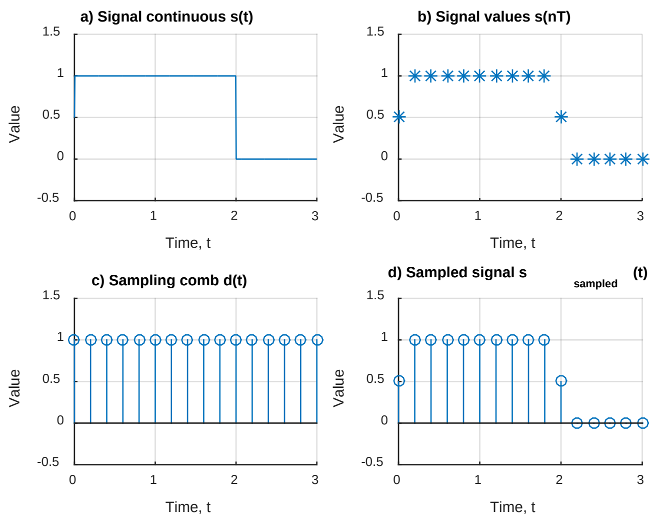


Fig. 2. Example of the sampled signal: the initial continuous signal (a), sampled values of the continuous signal (b), Dirac comb of delta functions - marked with vertical lines with a height equal to the area (c), sampled signal that is Dirac comb scaled with the samples (d). See Matlab code in Appendix B.

This is an idealized representation, because the delta functions are mathematical abstraction.

Some approximations can also be sequences of pulses (for example, short rectangular ones with a large voltage value), and the shorter their duration, the less distortions due to their shape will be introduced. This clearly illustrates the definition of the delta function as the limit of suitable pulses, which we used earlier [2].

Instead of infinite summation limits, finite ones can be used (if the original signal is time limited), for example, from 0 to N. In this case, the signal can be considered original, defined at infinity, multiplied by a function that takes values in the appropriate limits 1, and 0 is outside these limits.

B. *How the signal changes if only samples are used from it*

At first sight, we lose an infinite amount of information, because there were continuous intervals with an uncountable number of points, and only individual values remain.

However, under certain conditions, it is possible to restore the original continuous signal absolutely precisely from this limited number of points. In practice, even if these conditions are not strictly fulfilled, a fairly close result can be obtained.

To demonstrate that we could use spectral analysis using Fourier Transform [2]. The spectrum is one-to-one identifies the signal. So if the spectra of the two signals are the same, then the signals are the same.

Therefore, we consider what happens to the signal spectrum, if only samples have been selected from it.

We can write the function of a discrete signal as a product of the original continuous signal by a special function:

$$s_{sampled}(t) = s(t) \times \underbrace{\sum_{n=-\infty}^{\infty} \delta(t - nT)}_{d(t)}$$

This $d(t)$ function used for sampling is called the *Dirac comb*:

$$d(t) = \sum_{n=-\infty}^{\infty} \delta(t - T \times n), r \in Z$$

The **spectrum of the product** of two signals (Fourier transform) is the already known operation of *convolution* of the original signals (with some scaling):

$$s(t) = f(t)g(t) \leftrightarrow S(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} s(t)e^{-j\omega t}dt$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t)g(t)e^{-j\omega t}dt$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) \left[\underbrace{\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} G(u)e^{+jut}du}_{IFT} \right] e^{-j\omega t}dt$$
$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} G(u) \left[\underbrace{\int_{-\infty}^{\infty} f(t)e^{-j(\omega-u)t}dt}_{FT} \right] du$$
$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} G(u)F(\omega - u)du = \frac{1}{2\pi} \times (G * F)(\omega)$$

Therefore, if we select samples from the signal (using the Dirac Comb), the total spectrum will be a convolution of the original spectrum with the Comb spectrum.

What is the Dirac Comb spectrum:

$$D(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} d(t)e^{-j\omega t}dt$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - T \times n)\, e^{-j\omega t}dt$$
$$= \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t - T \times n)e^{-j\omega t}dt$$
$$= \underbrace{\frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} e^{-j\omega nT}}_{See\ Appendix\ A}$$
$$= \frac{\sqrt{2\pi}}{T} \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi}{T}n)$$

A detailed calculation of some steps (marked above in the formula) is given in Appendix A.

So, the Dirac comb in time (with the time step T and the scale 1) is a Dirac comb in frequencies space (with the frequency step $\frac{2\pi}{T}$ and the scale $\frac{\sqrt{2\pi}}{T}$).

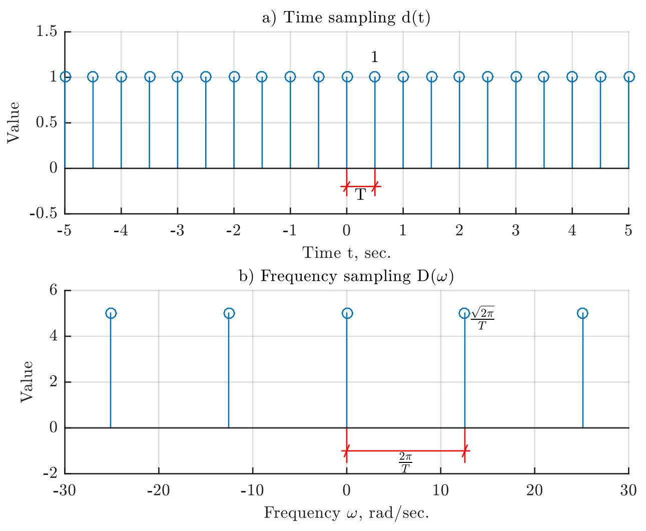
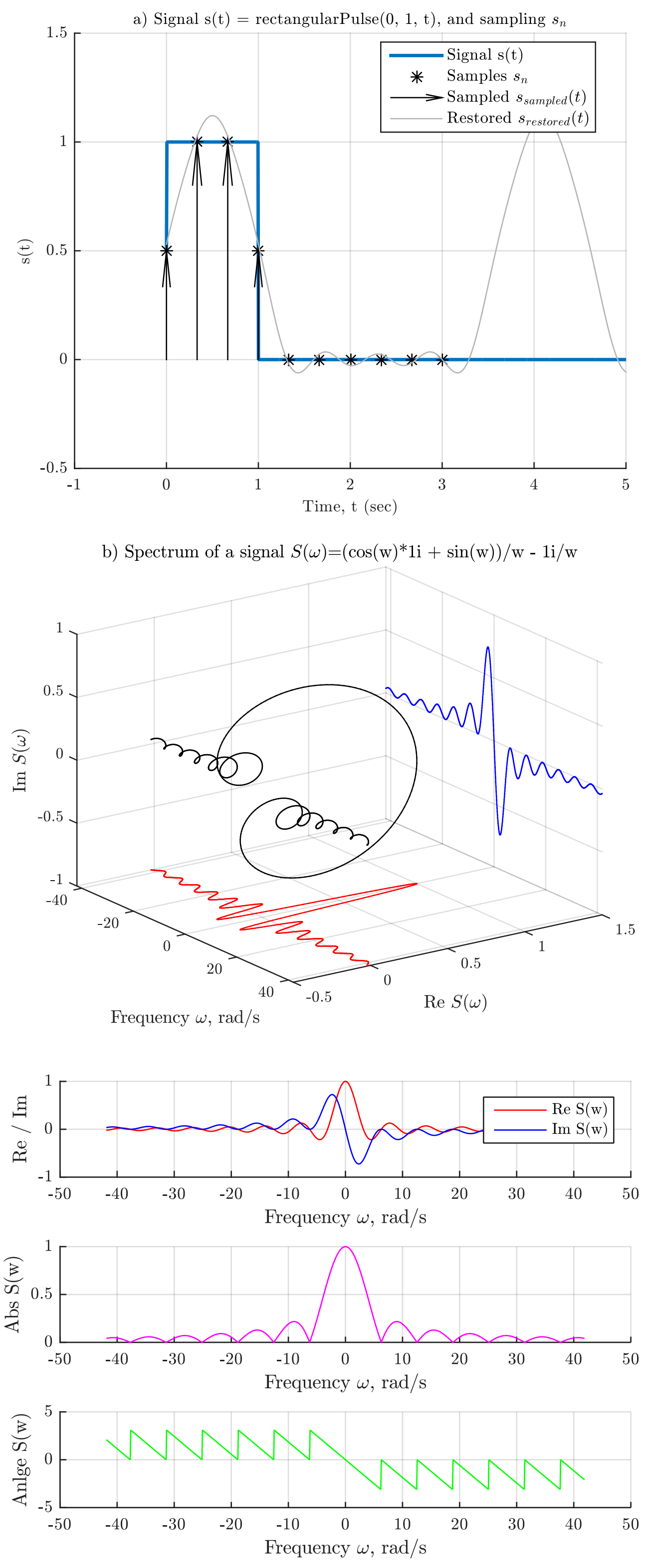


Fig. 3. Dirac comb in time space временном (a) and frequency space (b). See Matlab code in Appendix C.

Filtering property of the delta function means that convolution with it shifts the function to a given point. Therefore, the spectrum of the sampled signal is original continuous signal spectrum scaled by $\frac{1}{T\sqrt{2\pi}}$ and copied with a step of $\frac{2\pi}{T}$ equals to the sampling frequency :

$$S_{sampled}(\omega) = \frac{1}{2\pi} \times (S * D)(\omega)$$
$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} S(u)D(\omega - u)du$$
$$= \frac{1}{2\pi}$$
$$\times \frac{\sqrt{2\pi}}{T} \int_{-\infty}^{\infty} S(u) \sum_{n=-\infty}^{\infty} \delta(\omega - u$$
$$- \frac{2\pi}{T}n)\, du$$
$$= \frac{1}{T\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} S(u)\delta(\omega - u$$
$$- \frac{2\pi}{T}n)du$$
$$= \frac{1}{T\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} S\left(\omega - \frac{2\pi}{T}n\right) du$$



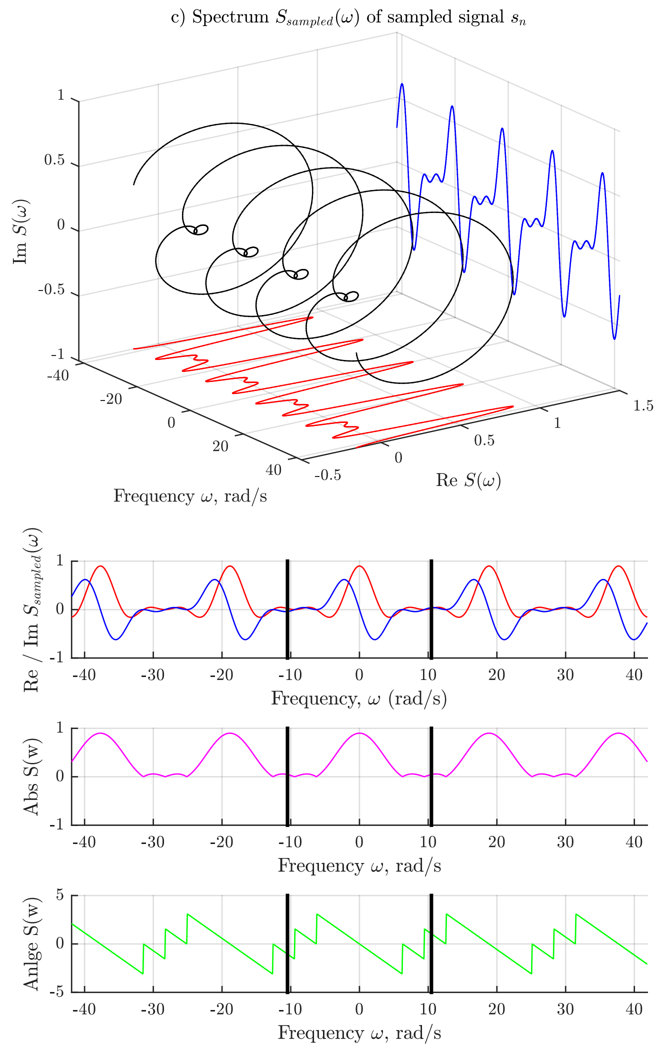


Fig. 4. Single rectangular pulse signal sampling: continuous signal, samples and sampled signal (a), spectrum of the original signal (b), spectrum of the sampled signal (c). See Matlab code in Appendix D.

IV. Kotelnikov Theorem: continuous signal recovery from the samples

As we see the spectrum of sampled (discretized) signal is different from the original continuous.

In the general case, the copies of a spectrum are superimposed on each other and distort so that it is impossible to obtain the original back (to restore the continuous signal).

However, for example, if the step between copies of the spectra (equal to the sampling frequency) is large enough compared to the width of the spectrum itself, they can be divided (by filtering all but one copy). Since the copies are located at a distance of the sampling frequency $\frac{2\pi}{T}$ from each other, the width of the spectrum of the continuous signal should **be 2 times smaller** than this step so that they do not overlap. This is the essence of Kotelnikov's theorem.

A. *Restoring with the ideal low pass filter*

To do this, we can apply an ideal low-pass filter (leaving only frequencies from 0 to Ω) or a band-pass filter (extracting the shifted spectrum from other copies), for example:

$$S_{restored}(\omega) = S_{sampled}(\omega) \times H_{lowpass}(w) \times T\sqrt{2\pi}$$

Where $H_{lowpass}$ is the transfer ratio (the value at the output of the filter divided by the value at the input):

$$H_{lowpass}(\omega) = \begin{cases} 1, & |\omega| < \Omega \\ 0, & |\omega| \geq \Omega \end{cases}$$

Impulse response (inversed Fourier transform) for it:

$$h(t) = \frac{1}{\sqrt{2\pi}} \int_{-\Omega}^{\Omega} e^{j\omega t} d\omega = \frac{1}{\sqrt{2\pi}} \frac{e^{j\Omega t} - e^{-j\Omega t}}{jt} = \frac{2\Omega}{\sqrt{2\pi}} \times \frac{\sin \Omega t}{\Omega t}$$

This filter is called “ideal low pass filter” or “sinc-filter” because of sinc-function ($sinc(x) = \frac{\sin x}{x}$). Unfortunately, it couldn’t be realized in practice – because it is infinite in time (in the future and in the past).

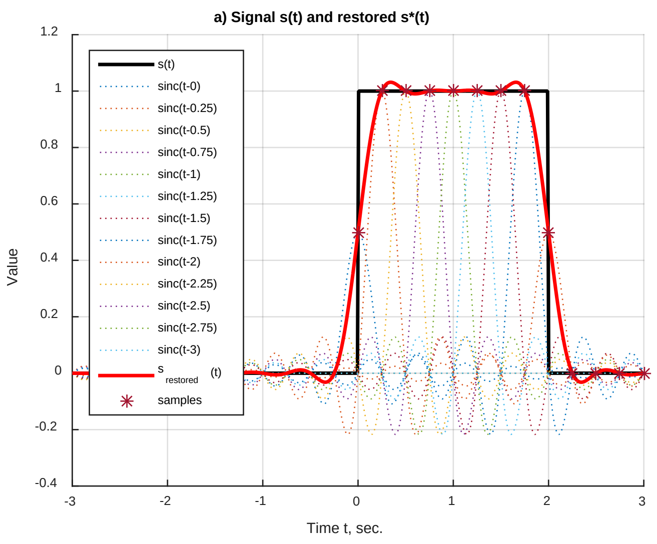
Spectra multiplication is equivalent to signal convolution:

$$S(\omega) = F(\omega)G(\omega) \leftrightarrow s(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} S(\omega) e^{j\omega t} d\omega$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega)G(\omega) e^{j\omega t} d\omega$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega) \left[\underbrace{\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\tau) e^{-j\omega\tau} d\tau}_{FT} \right] e^{j\omega t} d\omega$$
$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} g(\tau) \left[\underbrace{\int_{-\infty}^{\infty} F(\omega) e^{j\omega(t-\tau)} d\omega}_{IFT} \right] d\tau$$
$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} g(\tau) f(t-\tau) d\tau = \frac{1}{2\pi} (g * f)(t)$$

Thus, the signal restored by this frequency filtration:

$$s(t) = T\sqrt{2\pi} \times \frac{1}{2\pi} \int_{-\infty}^{\infty} s_{sampled}(\tau) h(t-\tau) d\tau$$
$$= \frac{T}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} s_n \delta(\tau - nT) \times \frac{2\Omega}{\sqrt{2\pi}} \times \frac{\sin \Omega(t-\tau)}{\Omega(t-\tau)} d\tau$$
$$= \frac{T\Omega}{\pi} \sum_{n=-\infty}^{\infty} s_n \int_{-\infty}^{\infty} \delta(\tau - nT) \frac{\sin \Omega(t-\tau)}{\Omega(t-\tau)} d\tau$$
$$= \frac{T\Omega}{\pi} \sum_{n=-\infty}^{\infty} s(nT) \frac{\sin(\Omega(t-nT))}{\Omega(t-nT)}$$

In order to avoid the imposition of spectra $T \leq \frac{\pi}{\Omega}$ is necessary. This is the mathematical formulation of the Kotelnikov theorem.



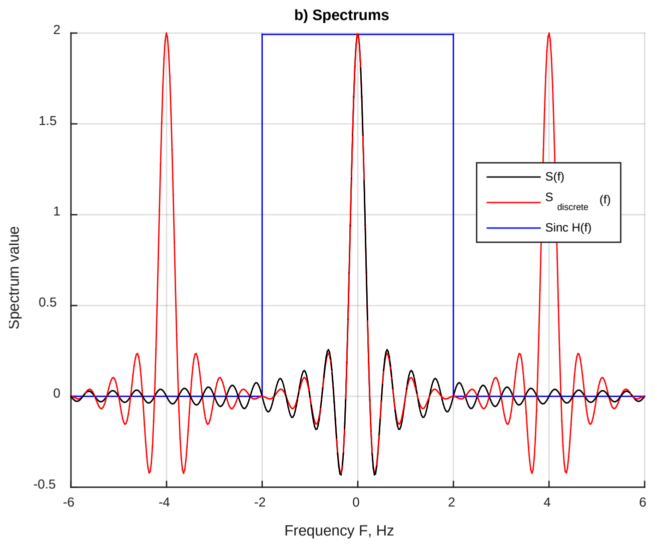


Fig. 5. The original continuous rectangular single pulse signal recovery from the samples: the original and reconstructed signal (a), the spectra of the original signal, the sampled signal and the filter (b). See Matlab code in Appendix E.

B. Restoring with not ideal low pass filter

An ideal low-pass filter (sinc-filter) is not physically realizable, since the sinc-function has non-zero values for all time points up to infinity, and the impulse response of an ideal filter is not zero for times less than zero. It can only be used mathematically.

For example, consider the signal restoration using the simplest low-pass filter - RC circuit from the article [1]

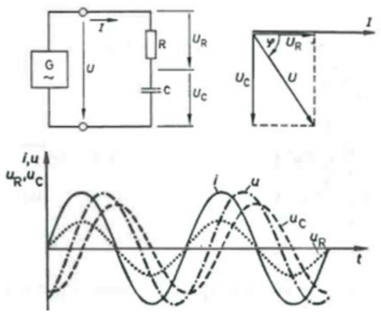


Fig. 6. RC circuit and its time and vector diagrams

The amplification spectrum (dependence of the output voltage, that is, the capacitor on the input voltage for each individual frequency of the input signal) depends on the impedances:

$$H(\omega) = \frac{U_C}{U_{in}} = \frac{Z_C}{Z_R + Z_C} = \frac{\frac{1}{j\omega C}}{R(1 + j\omega RC)} = \frac{1}{1 + j\omega RC}$$

Module (absolute value without phases) of this transfer characteristic:

$$|H(\omega)| == \frac{1}{\sqrt{1 + (\omega RC)^2}}$$

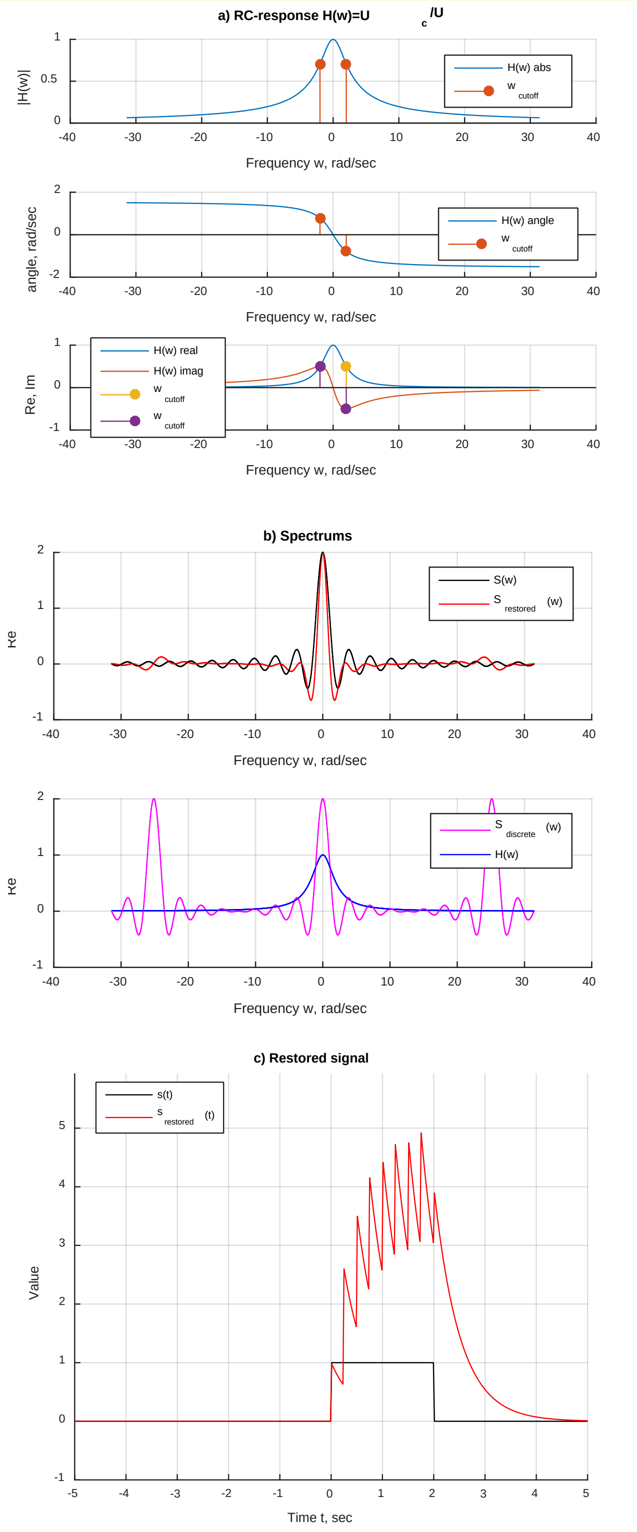


Fig. 7. The RC circuit can be used as a simple low pass filter: response spectrum (a) and a recovered sampled rectangular pulse signal (b). See Matlab code in Appendix F.

It can be seen that although RC-circuit is not very good low-pass filter. The level of power decreasing by a factor of 2 (that is, the voltage amplitude $\sqrt{2}$ times lesser) is achieved at a frequency $\Omega = \frac{1}{RC}$, this can be considered as a conditional filtering boundary.

For recovery, we can also directly simulate the physical processes in the circuit over time. As discussed in the first part of the article series the currents and voltages are related:

$$i_R(t) = \frac{E(t) - U_C(t)}{R} = i_C(t) = C\frac{dU_C(t)}{dt}$$

So state changing depends on the current state:

$$dU_C(t) = \frac{E(t) - U_C(t)}{RC} dt$$

We can assume the capacitor is initially discharged: $U_C(0) = 0$.

Of course, it is also necessary to replace the abstract delta functions, for example, with a pulse of finite duration and value (so that their product is single). Also, the simulation time will not be continuous, but divided into many small steps.

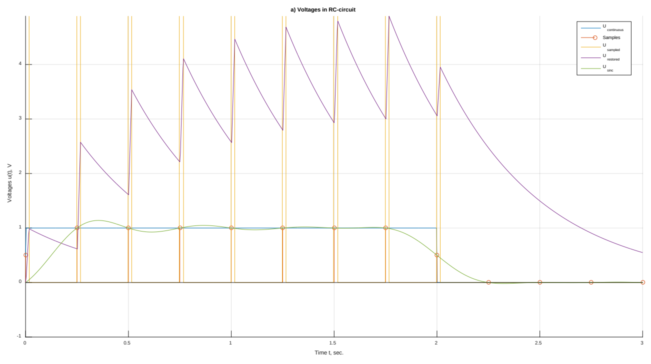


Fig. 8. Physical modeling of RC-circuit processing of a sampled signal sequence. See Matlab code in Appendix G.

The principle of operation of the RC-circuit is clearly seen: each pulse of the comb, due to its very large value, very quickly affects the charge of the capacitor (raising it to a significant level). After that, the capacitor discharges more slowly (because of its accumulated voltage is less than prompting pulse) until the next correction pulse, which charges it again, and so on.

Although the signal differs from the original, but it still reminds him.

In practice, much better low-pass filters are used, which give a result quite close to a sinc-filter.

V. Discrete Fourier Transform - Discrete spectrum for a finite number of samples

Summary, the spectrum of the sampled signal (peaks of the sequence of delta functions scaled to the samples values) is continuous, as is the original; and differs because of multiple copies at a distance corresponding to the time/sample rate.

From these samples, the original continuous signal can be accurately restored in some conditions.

But is it possible to discretize the spectrum so that it represents only a set of samples also? In this case, we again will not have to deal with continuous functions and we will be able to process them on digital devices.

The spectrum can be calculated from the readings directly:

$$S_{sampled}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} s_s(t) e^{-j\omega t} dt$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} s(t) \sum_{n=-\infty}^{\infty} \delta(t-nT)\, e^{-j\omega t} dt$$
$$= \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} s(t)\delta(t-nT) e^{-j\omega t} dt$$
$$= \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} s(nT) e^{-j\omega nT}$$
$$= \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} s_n e^{-j\omega nT}$$

If the number of available samples is finite (for example, from 0 to N – that we always have in real world), we respectively change the limits of summation:

$$S_{sampled}(\omega) = \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N} s_n e^{-j\omega nT}$$

Consider for this case the spectrum samples S_k at some selected frequency points $\omega = \frac{2\pi k}{NT}$:

$$S_k = S\left(\frac{2\pi k}{NT}\right) = \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N-1} s_n e^{-j\frac{2\pi}{NT}knT} = \sum_{n=0}^{N-1} s_n e^{-j\left(\frac{2\pi}{N}\right)kn}$$

They are interesting because of all the samples of the original sequence can be calculated only from them, without using all the other points and intervals of the continuous spectrum:

$$\sum_{k=0}^{N-1} S_k e^{j\frac{2\pi}{N}kn} = \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} s_m e^{-j\left(\frac{2\pi}{N}\right)km} \right) e^{j\frac{2\pi}{N}kn}$$
$$= \sum_{m=0}^{N-1} s_m \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(m-n)k} =$$
$$= \sum_{m=0}^{N-1} s_m \underbrace{\sum_{k=0}^{N-1} \underbrace{\left[e^{j\frac{2\pi}{N}(m-n)}\right]^k}_{a_k}}_{geometric\ progression\ sum}$$

Sum of N numbers (0..N-1) of geometrical progression a^k in last equation is:

$$\sum_{k=0}^{N-1} a^k = \begin{cases} \dfrac{1-a^N}{1-a}, & k \neq 1 \\ N, & k = 1 \end{cases}$$
$$= \begin{cases} \dfrac{1-e^{j\frac{2\pi}{N}(m-n)\times N}}{1-a}, & m \neq n \\ N, & m = n \end{cases}$$
$$= \begin{cases} \dfrac{\overbrace{1-\overbrace{e^{j2\pi\overbrace{(m-n)}^{integer}}}^{1}}^{0}}{1-a}, & m \neq n \\ N, & m = n \end{cases}$$
$$= \begin{cases} 0, & m \neq n \\ N, & m = n \end{cases}$$

So:

$$\sum_{k=0}^{N-1} S_k e^{j\frac{2\pi}{N}kn} = \sum_{m=0}^{N-1} s_m \times \begin{cases} 0, m \neq n \\ N, m = n \end{cases} = N \times s_n$$

This pair of one-to-one transformations that bind a finite number of samples of a discretized signal and a discretized spectrum is called the *Discrete Fourier transform* (DFT) or Discrete in time Fourier transform.

$$S_k = \sum_{n=0}^{N-1} s_n e^{-j\left(\frac{2\pi}{N}\right)kn}$$
$$s_n = \frac{1}{N} \sum_{k=0}^{N-1} S_k e^{j\frac{2\pi}{N}kn}$$

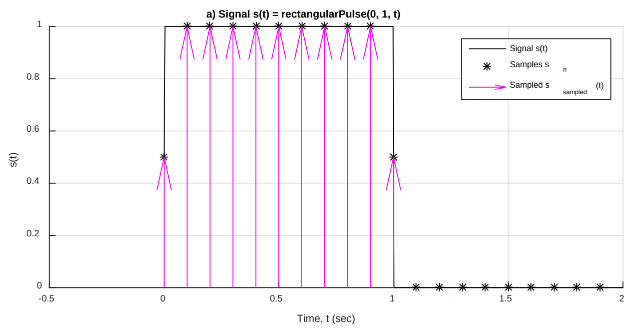
That is, we can use not the entire continuous spectrum, but only a small set of fixed points in it (number is equals to number of time samples), applying transformations and filters only to them.

At the same time, for real-valued signals only one half of the spectral samples is sufficient for restoring due to the conjugate symmetry (exact equality of the i-th and (N-1)-th reference):

$$S_{N-k} = \sum_{n=0}^{N-1} s_n e^{-j\left(\frac{2\pi}{N}\right)(N-n)k} = \sum_{n=0}^{N-1} s_n e^{-j2\pi k + j\left(\frac{2\pi}{N}\right)kn}$$
$$= \sum_{n=0}^{N-1} s_n e^{+j\left(\frac{2\pi}{N}\right)kn} = \mathrm{S_k}^*$$

Briefly Discrete Fourier transform also could be written in vector/matrix form:

$$S = \hat{F}s, \text{where } F_{m,n} = e^{-j\frac{2\pi}{N}(m-1)(n-1)}$$
$$S = \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ \dots \\ S_N \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-j\frac{2\pi}{N}\times 1} & e^{-j\frac{2\pi}{N}\times 2} & \dots & e^{-j\frac{2\pi}{N}\times(N-1)} \\ 1 & e^{-j\frac{2\pi}{N}\times 2} & e^{-j\frac{2\pi}{N}\times 8} & \dots & e^{-j\frac{2\pi}{N}\times 2(N-1)} \\ \dots & \dots & \dots & \ddots & \vdots \\ 1 & e^{-j\frac{2\pi}{N}\times(N-1)} & e^{-j\frac{2\pi}{N}\times 2(N-1)} & \dots & e^{-j\frac{2\pi}{N}\times(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \dots \\ s_N \end{pmatrix}$$



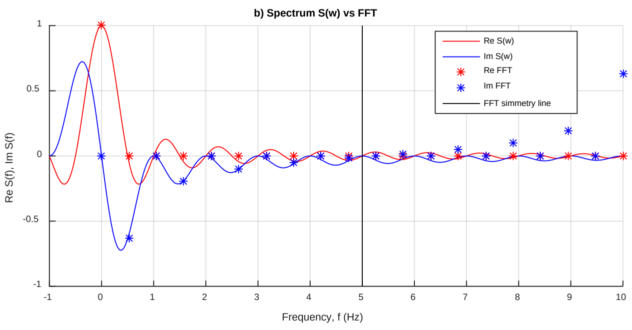


Fig. 9. Comparison of spectra: a continuous signal of a rectangular pulse and sampled spectrum frequencies of a sampled signal. Signal and its discretization shown at (a), and signal spectrum in comparison with DFT. One can see the symmetry (redundancy) of the DFT and the deviation of the points of the spectrum of the discretized signal from the line of the spectrum of the continuous signal due to the imposition of copies. See Matlab code in Appendix H.

There are algorithms that allow to find these spectral samples very quickly Fast Fourier Transform (FFT). The number of operations in them can be proportional to $N \log N$ for the number of samples N. They are implemented in various software products as standard functions [8].

VI. Conclusion

Thus, a continuous signal (if it's spectrum can be limited by acceptable bounds) can be turned into a set of samples, from which it can then be restored (if necessary or after processing). The number of samples directly depends on the selected boundaries of the used signal spectrum.

Moreover, for practical purposes the number of samples will be finite, which means that a finite number of samples of the sampled signal spectrum can be stored or processed.

Therefore, continuous signals of the real world can be turned into sets of numbers and processed by digital computing devices, as well as transmitted between them by any convenient means, including those with suitable modulation and at suitable frequencies.

That is the theme of the next part of article series.

Appendix

A. Dirac Comb spectrum calculations

$$D(\omega) = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} e^{-j\omega nT} = \left\{q_{1,2} = e^{\pm j\omega T}\right\}$$
$$= \frac{1}{\sqrt{2\pi}} \left[\underbrace{\sum_{n=0}^{\infty} {q_1}^n}_{\substack{Geometrical\\progression\ 1,\\Positive\ k}} + \underbrace{\sum_{n=0}^{\infty} {q_2}^{-n}}_{\substack{Geometrical\\progression\ 2,\\Negative\ k}} - \underbrace{1}_{\substack{Correspond\ to\\k=0}} \right]$$

Geometrical progression partial sum:

$$S_n = \sum_{k=0}^{n-1} q^k$$
$$= \begin{cases} n, & q = 1 \leftrightarrow e^{\pm j\omega T} = 1 \leftrightarrow \omega T = 2\pi r, \quad r \in \mathbb{Z} \\ \dfrac{1-q^n}{1-q}, & q \neq 1 \end{cases}$$

This sum sequence doesn't convergent in the usual sense because of infinite oscillating. In this cases under the limit could be understood Cesàro summation (as limit of average partial sum):

$$\sum_{k=1}^{\infty} a_n = A \overset{Cesàro}{\Longleftrightarrow} A = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} S_n = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \underbrace{\sum_{k=1}^{n} a_n}_{S_n}$$

For the case of discussed sequence:

$$\frac{1}{N}\sum_{n=1}^{N} S_n = \begin{cases} \dfrac{1}{N}\sum_{n=1}^{N} n == \dfrac{N\dfrac{(N+1)}{2}}{N} = \dfrac{N+1}{2}, & \omega T = 2\pi r \\ \dfrac{1}{N}\sum_{n=1}^{N} \dfrac{1-e^{\pm j\omega T n}}{1-e^{\pm j\omega T}}, & else \end{cases}$$

Separately operating for case $\omega T \neq 2\pi r$:

$$\frac{1}{N}\sum_{n=1}^{N} S_n = \frac{1}{N}\sum_{n=1}^{N} \frac{1-q^n}{1-q} = \frac{1}{N(1-q)}\left(N - \underbrace{\sum_{n=1}^{N} q^n}_{\substack{Geometrical\\progression\ 3}} \right)$$
$$= \frac{1}{N(1-q)}\left(N - \underbrace{q\frac{1-q^{N-1}}{1-q}}_{Partial\ sum\ of\ 3} \right)$$
$$= \frac{1}{1-q} - q\frac{1-q^{N-1}}{N(1-q)^2}$$

The limit for each progression is:

$$\lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} S_n$$
$$= \begin{cases} \lim\limits_{N\to\infty} \dfrac{N+1}{2} = \infty, & \omega T = 2\pi r \\ \dfrac{1}{(1-q)} - \underbrace{\lim_{N\to\infty} \dfrac{1}{N} \times \underbrace{\dfrac{\overbrace{1-q^{N-1}}^{<2}}{(1-q)^2}}_{Const\ \neq 0}}_{\to 0} = \dfrac{1}{(1-e^{\pm j\omega T})}, & else \end{cases}$$

The limit of sum of Geometrical progressions 1 and 2:

$$\lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} (S_n^1 + S_n^2) = \begin{cases} \lim\limits_{N\to\infty} (N+1) = \infty, & \omega T = 2\pi r \\ \dfrac{1}{(1-e^{j\omega T})} + \dfrac{1}{(1-e^{-j\omega T})}, & else \end{cases}$$

Separately operating for case $\omega T \neq 2\pi r$:

$$\frac{1}{(1-e^{j\omega T})}+\frac{1}{(1-e^{-j\omega T})}=\frac{(1-e^{-j\omega T})+(1-e^{j\omega T})}{(1-e^{j\omega T})(1-e^{-j\omega T})}$$
$$=\frac{2-(e^{j\omega T}+e^{-j\omega T})}{1-e^{-j\omega T}-e^{j\omega T}+\underbrace{e^{0}}_{1}}$$
$$=\frac{2-(e^{j\omega T}+e^{-j\omega T})}{2-(e^{j\omega T}+e^{-j\omega T})}=1$$

So,

$$D(\omega)=\frac{1}{\sqrt{2\pi}}\sum_{n=-\infty}^{\infty}e^{-j\omega nT}$$
$$=\frac{1}{\sqrt{2\pi}}\left[\lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N}(S_n^1+S_n^2)-1\right]$$
$$=\begin{cases}\infty, & \omega T=2\pi r\\ 0, & else\end{cases}$$

Let explore integral in some range around some point $\omega T=2\pi r, r\in Z$:

$$\int_{\frac{2\pi}{T}(r-1/2)}^{\frac{2\pi}{T}(r+1/2)} D(\omega)d\omega=\frac{1}{\sqrt{2\pi}}\int_{\frac{2\pi}{T}(r-1/2)}^{\frac{2\pi}{T}(r+1/2)}\left(\sum_{n=-\infty}^{\infty}e^{-j\omega nT}\right)d\omega$$
$$=\frac{1}{\sqrt{2\pi}}\int_{\frac{2\pi}{T}(r-1/2)}^{\frac{2\pi}{T}(r+1/2)}\left(\underbrace{\sum_{n=1}^{\infty}e^{j\omega nT}}_{Positive\ n}+\underbrace{\sum_{n=1}^{\infty}e^{-j\omega nT}}_{Negative\ n}\right.$$
$$\left.+\underbrace{1}_{\substack{Correspond\ to\\ n=0}}\right)d\omega$$
$$=\frac{1}{\sqrt{2\pi}}\sum_{n=1}^{\infty}\int_{\frac{2\pi}{T}(r+1/2)}^{\frac{2\pi}{T}(r+1/2)}(e^{j\omega nT}+e^{-j\omega nT})d\omega$$
$$+\frac{1}{\sqrt{2\pi}}\int_{\frac{2\pi}{T}(r+1/2)}^{\frac{2\pi}{T}(r+1/2)}d\omega$$
$$=\frac{1}{\sqrt{2\pi}}\sum_{n=1}^{\infty}\int_{\frac{2\pi}{T}(r-1/2)}^{\frac{2\pi}{T}(r+1/2)}\frac{\cos(\omega nT)}{2}d\omega+\frac{\frac{2\pi}{T}}{\sqrt{2\pi}}$$
$$=\frac{\sqrt{2\pi}}{T}$$
$$+\frac{1}{\sqrt{2\pi}}\sum_{n=1}^{\infty}\frac{1}{2nT}\left(\sin\left(\frac{2\pi}{T}(r+1/2)nT\right)\right.$$
$$\left.-\sin\left(\frac{2\pi}{T}(r-1/2)nT\right)\right)$$
$$=\frac{\sqrt{2\pi}}{T}$$
$$+\frac{1}{\sqrt{2\pi}}\sum_{k=1}^{\infty}\frac{1}{2nT}\left(\underbrace{\sin(2\pi nr+\pi n)}_{0}\right.$$
$$\left.-\underbrace{\sin(2\pi nr-\pi n)}_{0}\right)=\frac{\sqrt{2\pi}}{T}$$

That is, the area (integral) of the function in a neighborhood of points where it is nonzero is finite.

So, this is a scaled delta function:

$$D(\omega)=\frac{1}{\sqrt{2\pi}}\sum_{n=-\infty}^{\infty}e^{-j\omega nT}=\frac{\sqrt{2\pi}}{T}\sum_{n=-\infty}^{\infty}\delta\left(\omega-\frac{2\pi}{T}n\right)$$

B. *Matlab code: Sampling example (Figure 1)*

```
% Sampling singnal examle

%% Settings
s = @(t) rectangularPulse(0, 2, t); % signal func
Tmax = 3;           % max plot time
dt = 1/5;           % samplting time
accuracy = 500;     % continuous time steps

%% Processing
e = @(t) ones(size(t));                 % ones-vector
timeline = linspace(0, Tmax, 500);  % timeline to
plot
t_samples = 0:dt:Tmax;            % sampling timeline

%% Plotting
figure();

% plot signal
subplot(2, 2, 1); hold on; grid on;

title('a) Signal continuous s(t)');
xlabel('Time, t'); ylabel('Value');
axis([0 Tmax -0.5 1.5]);

plot(timeline, s(timeline), 'DisplayName', 's(t)
signal');

% plot signal
subplot(2, 2, 2); hold on; grid on;

title('b) Signal values s(nT)');
xlabel('Time, t'); ylabel('Value');
axis([0 Tmax -0.5 1.5]);

scatter(t_samples, s(t_samples), '*',
'DisplayName', 's(t) signal');

% plot comb
subplot(2, 2, 3); hold on; grid on;

title('c) Sampling comb d(t)');
xlabel('Time, t'); ylabel('Value');
axis([0 Tmax -0.5 1.5]);

stem(t_samples, e(t_samples), 'DisplayName',
'sampling');

% plot samples
subplot(2, 2, 4); hold on; grid on;

title('d) Sampled signal s_{sampled}(t)');
xlabel('Time, t'); ylabel('Value');
axis([0 Tmax -0.5 1.5]);

stem(t_samples, s(t_samples), 'DisplayName',
's_sampled');
```

C. *Matlab code: Dirac comb visualization (Figure 2)*

```
% Sampling comb visualization

%% Settings
Tmax = 5;               % max plot time, sec
Fmax = 5;               % max plot frequency, Hz
dt = 1/2;               % sampling time, sec

%% Processing
e = @(t) ones(size(t)); % ones-vector

Wmax = 2*pi*Fmax;       % max plot freq., rad/s
dw = 2*pi/dt;           % sampling freq., rad/s

N_t = floor(Tmax/dt);   % number of t-samples
N_w = floor(Wmax/dw);   % number of w-samples

t_samples = dt*[-N_t:N_t]; % sampling timeline
w_samples = dw*[-N_w:N_w]; % sampling freq.line
D_scale = sqrt(2*pi)/dt;   % frequency scaling

%% Plotting
figure();

% plot signal
subplot(2, 1, 1); hold on; grid on;
```

```
title('a) Time sampling d(t)');
xlabel('Time t, sec.'); ylabel('Value');
axis([-Tmax Tmax -0.5 1.5]);

stem(t_samples, e(t_samples), 'DisplayName', 'Time
comb d(t)');

text(dt, 1+0.1, '1', 'HorizontalAlignment',
'center', 'VerticalAlignment', 'bottom');
text(dt/2, -0.3, 'T', 'HorizontalAlignment',
'center');
line([0 0],[0 -0.3], 'Color', 'r', 'LineStyle','-
');
line([dt dt],[0 -0.3], 'Color', 'r',
'LineStyle','-');
line([-.1 dt+.1],[-.2 -.2], 'Color', 'r',
'LineStyle','-');
line([-.05 .05],[-.2-.05 -.2+.05], 'Color', 'r',
'LineStyle','-');
line([dt-.05 dt+.05],[-.2-.05 -.2+.05], 'Color',
'r', 'LineStyle','-');

% plot spectrum
subplot(2, 1, 2); hold on; grid on;

title('b) Frequency sampling D($\omega$)');
xlabel('Frequency $\omega$, rad/sec.');
ylabel('Value');

stem(w_samples, D_scale*e(w_samples),
'DisplayName', 'Frequency comb d(t)');

text(dw+0.4, D_scale-0.2,
'${\sqrt{2\pi}}\over{T}$');
text(dw/2, D_scale*-0.3, '${2\pi}\over{T}$',
'HorizontalAlignment', 'center');
line([0 0],[0 D_scale*-0.3], 'Color', 'r',
'LineStyle','-');
line([dw dw],[0 D_scale*-0.3], 'Color', 'r',
'LineStyle','-');
line([D_scale*-.1 dw+D_scale*.1],[D_scale*-.2
D_scale*-.2], 'Color', 'r', 'LineStyle','-');
line([D_scale*-.05 D_scale*.05],[D_scale*(-.2-.05)
D_scale*(-.2+.05)], 'Color', 'r', 'LineStyle','-
');
line([dw-D_scale*.05 dw+D_scale*.05],[D_scale*(-
.2-.05) D_scale*(-.2+.05)], 'Color', 'r',
'LineStyle','-');

% Interpreter for special symbols
set(groot,'defaulttextinterpreter','latex');
set(groot,'defaultAxesTickLabelInterpreter','latex
');
set(groot,'defaultLegendInterpreter','latex');
set(findall(gcf,'-
property','FontSize'),'FontSize',10)
```

D. Matlab code: sampled signal spectrum visualization (Figure 3)

```
%% Parameters
pulse_start = 0;
pulse_duration = 1;

N_samples = 10;          % number of samples

t_min=0;                 % min t of a signal
t_max=3;                 % max t of a signal


% sampling time
t_step = t_max/N_samples;
% sampled timeline
t_dicrete = linspace(t_min, t_max, N_samples);
% max time to plot signal
t_plot_max=5;
% timeline to plot signal
t_plot_range = linspace(t_min, t_plot_max,
N_samples*100);

% max frequency to plot
w_plot_max = 2*(2*pi)/t_step;
% min frequency to plot
w_plot_min = -w_plot_max;
% frequency line to plot
w_plot_range = w_plot_min:(w_plot_max-
w_plot_min)/1000:w_plot_max;
% zero-division fix
w_plot_range(find(w_plot_range==0))=0.001;

% max frequency for IFT
w_max = 2*pi/(2*t_step);
% min frequency for IFT
w_min = -w_max;
% sampling frequency
w_step = 2*pi/t_max;
% sampled frequent line
w_dicrete = linspace(w_min, w_max, N_samples+3);

% symbolic variables: time, frequency
syms t w;
symbolic function of s(t)
s_function =
rectangularPulse(pulse_start,pulse_duration,t);%


%% Processing
o = @(t) zeros(size(t));      % zero-vector
e = @(t) ones(size(t));       % ones-vector

% signal
s_continuous = eval(subs(s_function, t,
t_plot_range));
% samples
s_samples = eval(subs(s_function, t, t_dicrete));
% sampled signal
s_dicrete_function = sum((s_samples).*dirac(t-
t_dicrete));

% Spectrum – symbolic
S_function = fourier(s_function);
% Spectrum - array for plot
S_continuous = eval(subs(S_function, w,
w_plot_range));

% Spectrum discrete - symbolic
S_dicrete_function = fourier(s_dicrete_function);
% Spectrum discrete - array for plot normalized
S_dicrete = eval(subs(S_dicrete_function, w,
w_plot_range))*t_step;

s_restored_function = ifourier(S_disdis_function,
t);
s_restored = eval(subs(s_restored_function, t,
t_plot_range));

yscale=ceil(max(abs(S_dicrete)));

%% a) Signal continuous
figure(1); grid on; hold on;
axis([-1 t_plot_max -0.5 1.5]);

title(['a) Signal s(t) = ' char(s_function) ', and
sampling $s_n$']);
xlabel('Time, t (sec)');
ylabel('s(t)');

% Signal
plot(t_plot_range, s_continuous, 'LineWidth', 2,
'DisplayName', 'Signal s(t)');

% Sampling
scatter(t_dicrete, s_samples, '*', 'k',
'DisplayName', 'Samples $s_n$');
quiver(t_dicrete, o(t_dicrete), o(t_dicrete),
s_samples, 0, 'k', 'DisplayName', 'Sampled
$s_{sampled}(t)$');

% Restored
plot(t_plot_range, real(s_restored), 'Color', [.7
.7 .7], 'DisplayName', 'Restored
$s_{restored}(t)$');
```

```
legend('Location', 'best');

%% b) Signal spectrum -
% 3D
figure(2); grid on; hold on;
xlim([w_plot_min w_plot_max]);
view(55, 25);

title(['b) Spectrum of a signal $S(\omega)$='
char(S_function)]);
xlabel('Frequency $\omega$, rad/s');
ylabel('Re $S(\omega)$');
zlabel('Im $S(\omega)$');

plot3(w_plot_range, real(S_continuous),
imag(S_continuous), 'k');   % 3D curve
plot3(w_plot_range, real(S_continuous), -
e(w_plot_range), 'r');     % Re
plot3(w_plot_range, 1.5*e(w_plot_range),
imag(S_continuous), 'b');  % Im

% 2D
figure(3);
xlim([w_plot_min w_plot_max]);

subplot(3, 1, 1); grid on; hold on;

xlabel('Frequency $\omega$, rad/s');
ylabel('Re / Im');

plot(w_plot_range, real(S_continuous), 'r'); % Re
plot(w_plot_range, imag(S_continuous), 'b'); % Im

legend('Re S(w)', 'Im S(w)');

subplot(3, 1, 2); grid on; hold on;
xlabel('Frequency $\omega$, rad/s');
ylabel('Abs S(w)');
plot(w_plot_range, abs(S_continuous), 'm');  % Abs

subplot(3, 1, 3); grid on; hold on;
xlabel('Frequency $\omega$, rad/s');
ylabel('Anlge S(w)');

plot(w_plot_range, angle(S_continuous), 'g');% Ang

%% Spectrum of signal dicrete
figure(4);
grid on; hold on;

xlim([w_plot_min w_plot_max]);
view(55, 25);

title('c) Spectrum $S_{sampled}(\omega)$ of
sampled signal $s_n$');
xlabel('Frequency $\omega$, rad/s');
ylabel('Re $S(\omega)$');
zlabel('Im $S(\omega)$');

plot3(w_plot_range, real(S_dicrete),
imag(S_dicrete), 'k');   % 3D curve
plot3(w_plot_range, real(S_dicrete), -
e(w_plot_range), 'r');   % Re
plot3(w_plot_range, 1.5*e(w_plot_range),
imag(S_dicrete), 'b');   % Im

figure(5);

% Re - Im
subplot(3, 1, 1);
grid on; hold on;
xlim([w_plot_min w_plot_max]);
ylim([-yscale yscale]);

xlabel('Frequency, $\omega$ (rad/s)');
ylabel('Re / Im $S_{sampled}(\omega)$');

plot(w_plot_range, real(S_dicrete), 'r');
plot(w_plot_range, imag(S_dicrete), 'b');

line([2*pi/(2*t_step) 2*pi/(2*t_step)], [min(ylim)
max(ylim)], 'Color', 'k', 'LineWidth', 2);
line([-2*pi/(2*t_step) -2*pi/(2*t_step)],
[min(ylim) max(ylim)], 'Color', 'k', 'LineWidth',
2);

% Abs
subplot(3, 1, 2);
grid on; hold on;
xlim([w_plot_min w_plot_max]);
ylim([-yscale yscale]);

xlabel('Frequency $\omega$, rad/s');
ylabel('Abs S(w)');

plot(w_plot_range, abs(S_dicrete), 'm');   % Abs

line([2*pi/(2*t_step) 2*pi/(2*t_step)], [min(ylim)
max(ylim)], 'Color', 'k', 'LineWidth', 2);
line([-2*pi/(2*t_step) -2*pi/(2*t_step)],
[min(ylim) max(ylim)], 'Color', 'k', 'LineWidth',
2);

% Angle
subplot(3, 1, 3);
grid on; hold on;
xlim([w_plot_min w_plot_max]);

xlabel('Frequency $\omega$, rad/s');
ylabel('Anlge S(w)');

plot(w_plot_range, angle(S_dicrete), 'g');  % Ang.

line([2*pi/(2*t_step) 2*pi/(2*t_step)], [min(ylim)
max(ylim)], 'Color', 'k', 'LineWidth', 2);
line([-2*pi/(2*t_step) -2*pi/(2*t_step)],
[min(ylim) max(ylim)], 'Color', 'k', 'LineWidth',
2);


%% dicrete spectrum of disceted signal
figure(6);
grid on;hold on;
xlim([w_min w_max]);%ylim([-yscale yscale]);

title('d) Sampled spectrum of sampled signal');
xlabel('Frequency, w (rad/s)');
ylabel('Re Sd(w), Im Sd(w)');

% dicrete spectrum of dicrete function
S_samples = eval(subs(S_dicrete_function, w,
w_dicrete))*t_step;    % samples normalized
S_disdis_function = sum((S_samples).*dirac(w-
w_dicrete))*2;

% Re-Im
plot(w_plot_range, real(S_dicrete), 'Color', [1
0.7 0.7]);
plot(w_plot_range, imag(S_dicrete), 'Color', [0.7
0.7 1]);

quiver(w_dicrete, o(w_dicrete), o(w_dicrete),
real(S_samples), 0, 'r');
quiver(w_dicrete, o(w_dicrete), o(w_dicrete),
imag(S_samples), 0, 'b');

%% Design: total
set(groot,'defaulttextinterpreter','latex');
set(groot,'defaultAxesTickLabelInterpreter','latex
');
set(groot,'defaultLegendInterpreter','latex');
set(findall(gcf,'-
property','FontSize'),'FontSize',10)
```

E. Matlab code: signal restoring from the samples (Figure 4)

```
%% Parameters
syms t w;
```

```
s = rectangularPulse(0, 2, t); % signal sym func

Tmax = 3;                      % max plot time
Fc = 2;                        % sampling freq.

%% Processing
o = @(t) zeros(size(t));       % zero-vector

Wc = 2*pi*Fc;          % sampling angular freq.
dt = 1/2/Fc;           % sampling time step
dw = 1/dt;             % sampling frequency step
N = Tmax/dt;           % number of samples

Mmin = 0;              % samples start
Mmax = Tmax/dt;        % sampled finish

t_samples = dt*[Mmin:Mmax];  % sampling timeline

% plot timeline
t_plot = linspace(-Tmax, Tmax, 500);
% plot frequency-line
f_plot = linspace(-3*Fc, 3*Fc, 500);

% Spectrum - array for plot
Sw = eval(subs(fourier(s), w, 2*pi*f_plot));

% samples
s_samples = eval(subs(s, t, t_samples));
% sampled signal
s_dicrete_function = sum((s_samples).*dirac(t-
t_samples));

% Spectrum discrete - symbolic
S_dicrete_function = fourier(s_dicrete_function);
% Spectrum discrete - array for plot normalized
S_dicrete = eval(subs(S_dicrete_function, w,
2*pi*f_plot))*dt;

%% Spectrum
figure();
hold on; grid on;

title('b) Spectrums');
xlabel('Frequency F, Hz');
ylabel('Spectrum value');

plot3(f_plot, real(Sw), imag(Sw), 'k',
'DisplayName', 'S(f)');
plot3(f_plot, real(S_dicrete), imag(S_dicrete),
'r', 'DisplayName', 'S_{discrete}(f)');
plot([-3*Fc -Fc -Fc Fc Fc 3*Fc], [0 0
max(real(Sw)) max(real(Sw)) 0 0], 'b',
'DisplayName', 'Sinc H(f)');

legend('Location', 'best');

%% Signal
figure();
hold on; grid on;

title('a) Signal s(t) and restored s*(t)');
xlabel('Time t, sec.');
ylabel('Value');

plot(t_plot, eval(subs(s, t, t_plot)), 'k',
'LineWidth', 2, 'DisplayName', 's(t)'); % signal

x = o(t_plot);
for i=Mmin:Mmax
    f = eval(subs(s, t, dt*i))*sinc(Wc*(t_plot-
dt*i)/pi);
    x = x+f;
    plot(t_plot, f, ':', 'DisplayName',
strcat('sinc(t-', num2str(dt*i),')'));
end
plot(t_plot, x, 'r', 'LineWidth', 2,
'DisplayName', 's_{restored}(t)');

scatter(t_samples, eval(subs(s, t, t_samples)),
'*', 'DisplayName', 'samples');
```

```
legend('Location', 'northwest');
```

F. *Matlab code: RC-response spectrum (Figure 5)*

```
% RC response spectrum

%% Signal parameters
syms t;

s = rectangularPulse(0, 2, t); % signal sym. Func.
Tmax = 3;                      % max plot time
Fc = 2;                        % sampling freq.

%% Frequencies plot parameters
f0 = -5;              % show frequency from, Hz
f1 = 5;               % show frequency to, Hz
steps = 1000;         % frequency accuracy step

%% RC-circuit parameters
R = 100;              % R = 100 Om
C = 5e-3;             % C = 5 mF

%% Processing response

% frequency line
w = 2*pi*linspace(f0, f1, steps);
% cutoff frequency
w0 = 1/R/C;

% response function
H = @(w) 1./(1+j*w*R*C);
% response vector
H_C = H(w);

e = @(t) ones(size(t));              % zero-vector

%% Processing signal spectrum
t_plot = linspace(-Tmax, Tmax, 500); % timeline

dt = 1/2/Fc;                 % sampling time step
t_samples = dt*[0:Tmax/dt]; % sampling timeline

S_continuous = eval(fourier(s));% Spectrum array

s_samples = eval(subs(s, t, t_samples)); % samples
s_discrete_function = sum((s_samples).*dirac(t-
t_samples));  % sampled signal

% Spectrum discrete symbolic
S_discrete_function =
fourier(s_discrete_function);
% Spectrum discrete
S_discrete = eval(S_discrete_function)*dt;

%% Spectrum
figure();

subplot(2, 1, 1);
hold on; grid on;

title('b) Spectrums');
xlabel('Frequency w, rad/sec');
ylabel('Re');
zlabel('Im');

plot3(w, real(S_continuous), imag(S_continuous),
'k', 'DisplayName', 'S(w)');
plot3(w, real(H(w).*S_discrete),
imag(H(w).*S_discrete), 'r', 'DisplayName',
'S_{restored}(w)');

legend('Location', 'best');

subplot(2, 1, 2);
hold on; grid on;

xlabel('Frequency w, rad/sec');
ylabel('Re');
zlabel('Im');

plot3(w, real(S_discrete), imag(S_discrete), 'm',
'DisplayName', 'S_{discrete}(w)');
```

```
plot3(w, real(H(w)), imag(H(w)), 'b',
'DisplayName', 'H(w)');

legend('Location', 'best');

%% Plotting 2D
figure();

% Abs
subplot(3, 1, 1);
hold on; grid on;

title('a) RC-response H(w)=U_c/U');
xlabel('Frequency w, rad/sec');
ylabel('|H(w)|');

plot(w, abs(H_C), 'DisplayName', 'H(w) abs');
stem([-w0 w0], [power(2, -1/2) power(2, -1/2)],
'filled', 'DisplayName', 'w_{cutoff}');

legend('Location', 'best');

% Abs
subplot(3, 1, 2);
hold on; grid on;

xlabel('Frequency w, rad/sec');
ylabel('angle, rad/sec');

plot(w, angle(H_C), 'DisplayName', 'H(w) angle');
stem([-w0 w0], [angle(H(-w0)) angle(H(w0))],
'filled', 'DisplayName', 'w_{cutoff}');

legend('Location', 'best');

% Re, Im
subplot(3, 1, 3);
hold on; grid on;

xlabel('Frequency w, rad/sec');
ylabel('Re, Im');

plot(w, real(H_C), 'DisplayName', 'H(w) real');
plot(w, imag(H_C), 'DisplayName', 'H(w) imag');
stem([-w0 w0], [real(H(-w0)) real(H(w0))],
'filled', 'DisplayName', 'w_{cutoff}');
stem([-w0 w0], [imag(H(-w0)) imag(H(w0))],
'filled', 'DisplayName', 'w_{cutoff}');

legend('Location', 'best');
```

G. *Matlab code: RC-circuit signal restoring (Figure 6)*

```
%% Time parameters
T_start = 0;          % start time, sec
T_end = 3;            % finish time, sec

% continuous time accuracy step
dt_continuous = (T_end-T_start)/1000;

F_sampling = 2;       % sampling frequency, Hz

%% RC parameters
R = 100;             % 2 Om
C = 5e-3;            % 5 mF

%% Signal parameters
syms t;
U_symbolic = rectangularPulse(0, 2, t); % signal

delta_duration = dt_continuous*5; % delta-impulse
delta_value = 1/delta_duration;

%% Processing signal
% timeline
t_continuous = T_start:dt_continuous:T_end;
% null time vector
o = @(t) zeros(size(t));

dt_sampling = 1/2/F_sampling;
t_samples = T_start:dt_sampling:T_end;
N_samples = (T_end-T_start)/dt_sampling;

U_continuos = eval(subs(U_symbolic, t,
t_continuous));
U_samples = eval(subs(U_symbolic, t, t_samples));

%% Real delta-output
U_out = o(t_continuous);
% Scaled delta grid
for i=0:N_samples
    current_time = i*dt_sampling;
    U_sample = eval(subs(U_symbolic, t,
current_time));
    U_out = U_out +
U_sample*delta_value*rectangularPulse(current_time
, current_time+delta_duration, t_continuous);
end

%% Processing CR-circuit: step-by-step modeling
U_C = (0);
for i = 1:size(U_out, 2)-1
    dU_C = (U_out(i)-U_C(i))/R/C*dt_continuous;
    U_C(i+1) = U_C(i)+dU_C;
end

%% Processing sinc-filter
% W_sampling = 2*pi*F_sampling;
U_LPF = (0);
for i=1:size(U_out, 2)
    current_time = i/2/F_sampling;
    f = eval(subs(U_symbolic, t, current_time)) *
sinc(2*F_sampling*(t_continuous-current_time));
    U_LPF = U_LPF + f;
end

%% Plot a: RC modeling voltage
figure();

axis([T_start T_end -1 max(U_C)+1]);
% subplot(2, 1, 1);
hold on; grid on;
title('a) Voltages in RC-circuit');
xlabel('Time t, sec.'); ylabel('Voltages u(t),
V');

plot(t_continuous, U_continuos, 'DisplayName',
'U_{continuous}');
stem(t_samples, U_samples, 'DisplayName',
'Samples');
plot(t_continuous, U_out, 'DisplayName',
'U_{sampled}');
plot(t_continuous, U_C, 'DisplayName',
'U_{restored}');                            % plot
voltage
plot(t_continuous, U_LPF, 'DisplayName',
'U_{sinc}');                              % plot
voltage

legend('Location', 'best');
```

H. *Matlab code: comparing continuous spectrum and DFT (Figure 7)*

```
figure('units','normalized','outerposition',[0    0
0.5 1]);

%% Parameters

%Signal
syms t;                 % t - symbolic variable
% symbolic signal t
s_function = rectangularPulse(0, 1, t);
% Symbolic expression, Fourier function
S_function = fourier(s_function);

% Time parameters
N_counts = 20;          % number of samples
N_plot = 500;           % plot accuracy
```

```
Tmax=2;              % plotting start time
Tmin=0;              % plotting end time

%% Processing

% sampling time step
dt = (Tmax-Tmin)/N_counts;
% plot timeline
t_plot = linspace(Tmin, Tmax, N_plot);
% sampling timeline
t_samples = dt*(0:N_counts-1);

o = @(t) zeros(size(t));      % zero-vector

%% Fourier processing

% Plot frequency line
f_plot = linspace(-1/dt, 1/dt, N_plot);
f_plot(f_plot==0)=0.0001;     % zero-division fix

% Sampling frequencies
f_samples = linspace(0, 1/dt, N_counts);
f_samples(f_samples==0)=0.0001;% zero-disivion fix

% Evaluate symbolics spectrums

% Continuous
w = 2*pi*f_plot;
S_continuous = eval(S_function);

% Spectrum samples
w = 2*pi*f_samples;
S_samples = eval(S_function);

% Signal sampled
s_sampled = eval(subs(s_function, t, t_samples));

%% Signal plot

subplot(2, 1, 1);
grid on; hold on;

title(['a) Signal s(t) = ' char(s_function)]);
xlabel('Time, t (sec)');
ylabel('s(t)');

% signal
plot(t_plot,  eval(subs(s_function,  t,  t_plot)),
'k');
% samples
scatter(t_samples, s_sampled, '*', 'k');
% sampled signal
quiver(t_samples,    o(t_samples),    o(t_samples),
s_sampled, 0, 'm');

legend('Signal  s(t)',  'Samples  s_n',  'Sampled
s_{sampled}(t)');

%% Plot spectrums
subplot(2, 1, 2);
grid on; hold on;

xlim([-1 1/dt]);    % limited axes

title('b) Spectrum S(w) vs FFT');
xlabel('Frequency, f (Hz)');
ylabel('Re S(f), Im S(f)');

% Spectrum
plot(f_plot, real(S_continuous), 'r');
plot(f_plot, imag(S_continuous), 'b');

% FFT plotting
fft_S = fft(s_sampled)/N_counts*2; % Normalization

scatter(f_samples, real(fft_S), '*', 'r'); % Re
scatter(f_samples, imag(fft_S), '*', 'b'); % Im

% Symmetry line
plot([1/2/dt 1/2/dt], [-1 1], 'k');

legend('Re  S(w)',  'Im  S(w)',  'Re  FFT',  'Im  FFT',
'FFT simmetry line');
```

References

[1] E. Tikhonov, M. Sneps-Sneppe. “Introduction to signal processing: sine wave and complex signals”, International Journal of Open Information Technologies, Vol. 7, No 3, March 2019, ISSN 2307-8162.

[2] E. Tikhonov, M. Sneps-Sneppe. “Introduction to signal processing: spectral representation”, International Journal of Open Information Technologies, Vol. 7, No 4, April 2019, ISSN 2307-8162.

[3] Lyons, Richard G., “Understanding digital signal processing”, 3rd ed., Library of Congress Cataloging-in-Publication Data, 1948, ISBN 0-13-702741-9

[4] Kotelnikov VA, “On the transmission capacity of "ether" and wire in electrocommunications”, (English translation, PDF, url: http://ict.open.ac.uk/classics/1.pdf), Izd. Red. Upr. Svyazzi RKKA (1933), Reprint in Modern Sampling Theory: Mathematics and Applications, Editors: J. J. Benedetto und PJSG Ferreira, Birkhauser (Boston) 2000, ISBN 0-8176-4023-1

[5] Robert J. Marks II, Introduction to Shannon Sampling and Interpolation Theory, Springer-Verlag, New York, 1991. Available: https://marksmannet.com/RobertMarks/REPRINTS/1999_IntroductionToShannonSamplingAndInterpolationTheory.pdf. Retrieved: May, 2019

[6] Kharkevich A.A., “Essays on the general theory of communication” (Russian), State publishing house of technical and theoretical literature, Moscow, 1955

[7] Ilyin V.A., Sadovnichy V.A, Sendov Bl.H. "Real Numbers // Mathematical Analysis" (Russian), 3rd ed. pererab. and add. / Ed. Tikhonov A. N., Prospect, Moscow 2006, ISBN 5-482-00445-7

[8] Emmanuel C. Ifeachor (2011) “Digital Signal Processing A practical approach download”. Prentice Hall; 2 edition (October 7, 2001), ISBN-10: 9780201596199