Introduction to signal processing: information transmission using orthogonal sine waves

E. Tikhonov, M. Sneps-Sneppe

***Abstract* — Data transmission using orthogonal functions means combining base carrier signals (scaled by numbers of useful information) into one sent signal. Processing the mixed signal after receiving allows dividing it back into carriers and restoring useful numbers.**

What is the most compact set of complex sine waves for this? How will the random components of the spectrum between the system signals (or the imperfection of the parameters) affect the result? What is the algorithm for processing the received signal and what is the minimum time needed to obtain the initial information? These issues are discussed "from the bottom to up", from simple physical considerations to mathematical expressions and processing algorithms in the MatLab package.

***Keywords* — scalar product, sine waves, complex sine waves, complex signals, Fourier series, Discrete Fourier Transform, MatLab.**

I. Introduction

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

In previous papers, complex signal representation [1], spectral analysis principles [2] and sampling [3] were discussed. The current article is dedicated to a method sending information (flow of numbers) using sets of orthogonal sine waves.

The rest of the paper is the following. Section 2 illustrates the physical principle of composing and decomposing a signal into base components, Section 3 presents the operation of the scalar product of functions. In Section 4 and Section 5, the method is applied to real and complex sine wave systems respectively. The Appendix contains the MatLab codes (used as the examples and produced illustration figures to all parts) for version MatLab 6.5 (R2015a).

II. Signal mix

The process of decomposition the signal into spectral components, considered in the paper [2] is reversible: the total signal can be obtained by summing various sine waves back. The use of the orthogonal base signal systems is similar, but there is a finite set of suitable reference components instead of a continuous spectrum. To decompose them and extract the encoded information in the receiver, a special tool should be used - the scalar product of functions.

This is illustrated in Fig. 1, which demonstrates the reconstruction of a signal from individual components (modified picture from [2] and [4]).

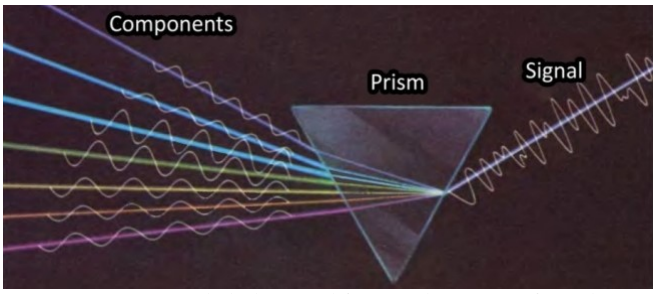

Fig. 1. A single light signal forming by a prism from the components

III. Scalar product of signals

In two-dimensional geometry there is a useful tool - the **scalar product** of vectors. It may be written in the form of the product of vector lengths by the cosine of the angle between them: $\langle \vec{f}, \vec{g} \rangle = |f||g| \cos \alpha$, or in the form of Cartesian coordinates [5]: $\langle \vec{f}, \vec{g} \rangle = f_x g_x + f_y g_y$. Scalar product characterizes the similarity or mutual (in)dependence of these vectors: it is 0 when they are perpendicular. Another important property of the scalar product is that it reflects length (norm) of the vector: $\langle \vec{f}, \vec{f} \rangle = |f|^2 = f \times f$.

The definition may be expanded with the preservation of its useful properties. For high-dimension geometry (with an arbitrary number of coordinates N) an extended definition of the coordinate form is: $\langle \vec{f}, \vec{g} \rangle = \sum_{i=1}^{N} f_i g_i$. In the area of complex numbers its definition is refined: $\langle \vec{f}, \vec{g} \rangle = \sum_{i=1}^{N} f_i g_i^*$ (the second vector coordinates are replaced with the conjugate [1]). This is done in order to preserve the property of square norm ($|f|^2 = f \times f^*$). Since the complex conjugation of a real number is the same real number, this form is correct for all vectors, both real and complex.

This definition is generalized for the case of **continuous complex functions** on a given parameter interval L. It is done as if the coordinates of the vector correspond to infinite number of infinitely small intervals, into which this interval is divided. Replacing the sum by the integral gives [5]:

$$\langle f, g \rangle_{\mathrm{L}} = \langle f(t), g(t) \rangle_L \stackrel{\text{def}}{=} \int_{t_0}^{t_0+L} f(t) \times g^*(t)\, dt \qquad (1)$$

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).

Where $f(t)$ and $g(t)$are multiplied functions, the time t_0 corresponds to the beginning, and the time L corresponds to the duration of the product calculation.

Objects with zero scalar product are called **orthogonal** (for vectors this corresponds to perpendicularity). They are independent of each other in some senses (different for different types of multiplied objects).

Scalar product has the property of **linearity** (for all cases): $\langle A \times f + B \times g, h\rangle = A\langle f, h\rangle + B\langle g, h\rangle$, where A and B are numbers and f, g, h – multiplied objects (numbers, vectors, functions and etc.). Therefore, it is easy to restore all the coefficients (representing useful information) of a signal that is a linear combination of orthogonal (base) functions (with a help of the scalar product of this signal with these base functions).

For example, if there are two orthogonal functions f and g (such that $\langle f, g\rangle_L = 0$), constant numbers A, B and signal $s(t) = A \times f(t) + B \times g(t)$:

$$\begin{aligned}\langle s, f\rangle_L &= \langle A \times f(t) + B \times g(t), f(t)\rangle_L \\ &= A\langle f, f\rangle_L + B\underbrace{\langle g, f\rangle_L}_{=0} = A\langle f, f\rangle_L \\ \Leftrightarrow A &= \frac{\langle s, f\rangle_L}{\langle f, f\rangle_L}\end{aligned} \quad (2)$$

IV. Real sine waves

A. Orthogonality of two sine waves

Real sine waves can be written in mathematical form [1]:

$$\begin{cases} f(t) = \cos(\omega_1 t + \varphi_1) \\ g(t) = \cos(\omega_2 t + \varphi_2) \end{cases} \quad (3)$$

So their scalar product will be determined by the formula (using designations $\omega^+ = \omega_1 + \omega_2$; $\omega^- = \omega_1 - \omega_2$; $\varphi^+ = \varphi_1 + \varphi_2$; $\varphi^- = \varphi_1 - \varphi_2$):

$$\begin{aligned}\langle f, g\rangle_{\mathrm{L}} &= \int_{t_0}^{t_0+L} \cos(\omega_1 t + \varphi_1)\cos(\omega_2 t + \varphi_2)\, dt \\ &= \frac{1}{2}\int_{t_0}^{t_0+L} [\cos(\omega^+ t + \varphi^+) \\ &\quad + \cos(\omega^- t + \varphi^-)]dt\end{aligned} \quad (4)$$

These signals will be orthogonal on those calculation time intervals L, where their scalar product becomes zero. Consider all such possibilities for different initial cases.

1) Trivial case ($\omega^+ = 0$; $\omega^- = 0 \leftrightarrow \omega_1 = \omega_2 = 0$):

$$\begin{aligned}\langle f, g\rangle_{\mathrm{L}} &= \frac{1}{2}\int_{t_0}^{t_0+L} [\cos(\varphi^+) + \cos(\varphi^-)]dt \\ &= \frac{L}{2}[\cos(\varphi^+) + \cos(\varphi^-)] \\ &= const\end{aligned} \quad (5)$$

In this case, the base signals are constant, therefore either everywhere is orthogonal, or not everywhere. The case is obviously useless for transmitting information.

2) Case of equal/opposite frequencies

This is the case of equal frequencies ($\omega_1 = \omega_2 = \omega \neq 0 \leftrightarrow \omega^+ \neq 0$; $\omega^- = 0$) and reverse case of opposite frequencies ($\omega^- \neq 0$; $\omega^+ = 0$), leading to exactly the same results:

$$\begin{aligned}\langle f, g\rangle_{\mathrm{L}} &= \frac{1}{2}\int_{t_0}^{t_0+L} [\cos(2\omega t + \varphi^+) + \cos(\varphi^-)]dt \\ &= \underbrace{\frac{\cos\varphi^-}{2}}_{Linear} L + \underbrace{\frac{1}{2\omega}\sin\omega L \cos(\omega L + [2\omega t_0 + \varphi^+])}_{Harmonic}\end{aligned} \quad (6)$$

The result is the sum of the linearly increasing (with calculation time L) summand $\frac{\cos\varphi^-}{2}L$ and the product of two sinusoids (scaled by $\frac{1}{2\omega}$). Since each sinusoid value is limited to $[-1; 1]$, the second summand is in range $\left[-\frac{1}{2\omega}; \frac{1}{2\omega}\right]$ always. After a while the first summand will exceed it, so there will be no stable zeros. That is, functions may be orthogonal (i.e. zeros exist and depend on the initial parameters, in particular, the initial phases at the moment of the beginning of the summation) only for time lesser than $\frac{1}{\omega L\cos(\varphi^-)}$. For sufficiently long transmissions such functions are not suitable.

The important case of zero linear growth ($\cos\varphi^- = 0$, i.e. $\varphi^- = \varphi_1 - \varphi_2 = \frac{\pi}{2} + \pi n,\ n \in \mathbb{Z}$) is an exception. In this case, the phases of the original signals must differ by $\frac{\pi}{2}$. That is, the signals are a sine and a cosine waves on the same frequency:

$$\begin{cases} f(t) = \pm\sin(\omega t + \varphi) \\ g(t) = \pm\cos(\omega t + \varphi) \end{cases} \quad (7)$$

The scalar product zeroes (orthogonality intervals) in this case always exist at points that doesn't depend on the initial phases and the beginning of the summation moment: $\sin\omega L = 0$. They correspond to calculation time intervals:

$$L = \frac{\pi}{\omega}n, n \in \mathbb{Z} \quad (8)$$

Thus, at the same time we can transmit **2 numbers on the same frequency**. In particular, these numbers can be interpreted as components of a two-dimensional vector. Or as parts of a complex number (real and imaginary components, or angle and length). In this sense, we can say that **at each frequency one complex number** can be transmitted.

3) Common case of different frequencies ($\omega^+ \neq 0, \omega^- \neq 0$)

$$\begin{aligned}\langle f, g\rangle_{\mathrm{L}} &= \frac{1}{2}\int_{t_0}^{t_0+L} [\cos(\omega^+ t + \varphi^+) + \cos(\omega^- t + \varphi^-)]dt \\ &= \frac{L}{2}\left[\frac{\sin\frac{\omega^+ L}{2}}{\frac{\omega^+ L}{2}}\cos\left(\frac{\omega^+(2t_0 + L) + 2\varphi^+}{2}\right)\right. \\ &\quad \left. + \frac{\sin\frac{\omega^- L}{2}}{\frac{\omega^- L}{2}}\cos\left(\frac{\omega^-(2t_0 + L) + 2\varphi^-}{2}\right)\right] \\ &= \frac{L}{2}[\operatorname{sinc} x\cos(x + u) + \operatorname{sinc} y\cos(y + v)]\end{aligned} \quad (9)$$

Here the function $\operatorname{sinc} x = \begin{cases} \frac{\sin x}{x}, & x \neq 0 \\ 1, & x = 0 \end{cases}$ and variables $x = \frac{\omega^+ L}{2}$; $y = \frac{\omega^- L}{2}$; $u = \omega^+ t_0 + \varphi^+$; $v = \omega^- t_0 + \varphi^-$ are used.

Fig. 2 shows example of calculation time dependence for the scalar products for signals $s_1(t) = \cos(\pi t)$ and $s_2(t) =$

$\cos(\frac{2\pi}{3}t+\frac{\pi}{2})$ by definition (4), realized by MatLab program, see Appendix A and B.

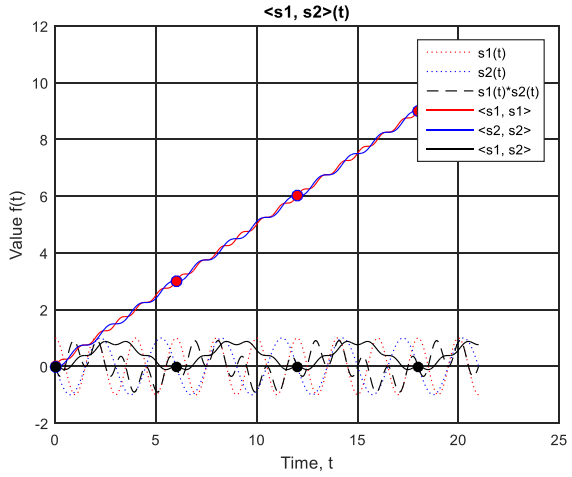

Fig. 2. Signals and scalar product results in time for two sine waves

Points of orthogonality ($\langle f,g\rangle_L = 0$) where function do not affect each other correspond to:

$$\operatorname{sinc} x \cos(x+u) = -\operatorname{sinc} y \cos(y+v) \qquad (10)$$

The parameters u and v depend on the usually unknown random phases at the time of the beginning of the recovery (summation) and should not be relied upon. Although they may in some cases give additional time points at which the base functions will be mutually orthogonal (for example, where one or both of the cosines turn to 0), they will disappear even with small changes in the initial phase.

Fig. 3 shows surface for (10) with zeros marked as peaks (see Appendix C) with random initial phases.

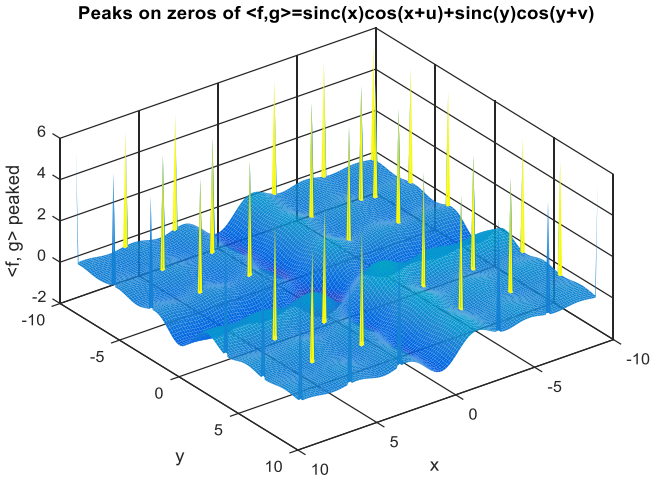

Fig. 3. Plane of $\langle f,g\rangle$ possible values with peaks on zeros.

Therefore, the stable solution is at points (taking into account the case in question: $\omega^+ \neq 0, \omega^- \neq 0$):

$$\begin{cases}\operatorname{sinc} x = 0\\ \operatorname{sinc} y = 0\end{cases} \leftrightarrow \begin{cases} x = \dfrac{\omega^+ L}{2} = \pi n,\\ y = \dfrac{\omega^- L}{2} = \pi k,\end{cases}$$
$$\leftrightarrow \begin{cases}\omega^+ = \omega_1 + \omega_2 = \dfrac{2\pi}{L}n, & n \neq 0\\ \omega^- = \omega_1 - \omega_2 = \dfrac{2\pi}{L}k, & k \neq 0\end{cases} \qquad (11)$$

So **all possible pairs** of the orthogonal frequencies on the calculating time L are:

$$\begin{cases}\omega_1 = \dfrac{\pi}{L}(n+k)\\ \omega_2 = \dfrac{\pi}{L}(n-k)\\ n,k \in \mathbb{Z}; n,k \neq 0\end{cases} \leftrightarrow \begin{cases} f(t) = \cos\left(\dfrac{\pi}{L}(n+k)+\varphi_1\right)\\ g(t) = \cos\left(\dfrac{\pi}{L}(n-k)+\varphi_2\right)\\ n,k \in \mathbb{Z}; n,k \neq 0\end{cases} \qquad (12)$$

Note, that at each of these frequencies, two basis signals (sine and cosine) can be used simultaneously in accordance with case 2.

B. Orthogonal real sine waves system

By the system of orthogonal waves we mean the set of base sine waves that do not affect each other at certain moments of the calculation (the same for all of them).

As was shown in (12) following frequencies (and only them) will be orthogonal to each other on the time interval L:

$$\begin{cases}\omega_2 = \dfrac{\pi}{L}(n-k)\\ \omega_1 = \dfrac{\pi}{L}(n+k)\\ n,k \in \mathbb{Z};\ n,k \neq 0\end{cases} = \begin{Bmatrix} r = n-k\\ \Delta\omega = \dfrac{2\pi}{L}\end{Bmatrix}$$
$$= \begin{cases}\omega_2 = \dfrac{\pi}{L}r = \Delta\omega \times {}^{r}/_{2}\\ \omega_1 = \omega_2 + \dfrac{2\pi}{L}k = \Delta\omega \times \left({}^{r}/_{2}+k\right)\\ r,k \in \mathbb{Z};\ k \neq \{-r,0\}\end{cases} \qquad (13)$$

It can be seen that the solution of this system for each L is divided into two independent sets of allowable frequencies $\{\omega_n\}_{odd}$ and $\{\omega_n\}_{even}$ (each element in the set may correspond to its own ω_1 or ω_2; with a step $\Delta\omega = \frac{2\pi}{L}$ between the nearby elements of the set). The first of them (odd) corresponds to $r = 1$, the second (even) to $r = 2$.

In each set, any pair of frequencies orthogonal if their indexes n do not coincide and are not opposite. Therefore, the selection of any element from the set for a system of basic functions (for example, with the index $n = 1$) excludes for further use as the basic signal this element and its opposite (in current example, indexes $n = 1$ and $n = -1$). The remaining elements (for example, with indexes $n = \pm 2,3,4,\ldots$) remain available for further selection (for example, the next frequency in the system may correspond to $n = -2$). Therefore, for any index value $|n|$, we can choose only one arbitrary sign of the frequency (positive or negative).

Therefore, we obtain **two series of orthogonal real sine wave systems**: "even" and "odd":

$$\{\omega_n\}_{even} = \left\{\pm_{\mathrm{n}}\frac{2\pi}{L}n, n \geq 0\right\}$$
$$\{\omega_n\}_{odd} = \left\{\pm_{\mathrm{n}}\frac{2\pi}{L}\left(n+\frac{1}{2}\right), n \geq 0\right\} \qquad (14)$$

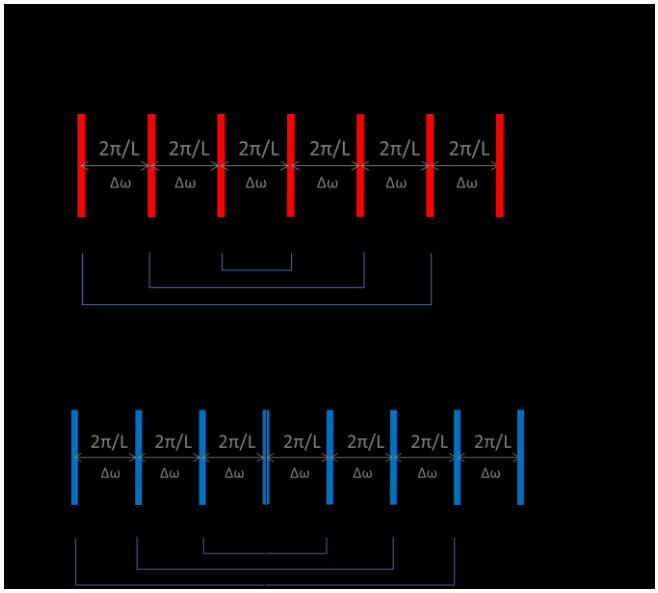

Fig. 4. Two systems (series) of real orthogonal sine waves corresponding to restore time L

The "even" series with all positive frequencies is a classical **Fourier series**, which is usually proposed as a solution for orthogonal systems. The second is an **additional one** (with a frequency shift $-\frac{\pi}{L}$ from the Fourier series). Also, instead of any frequency **equal opposite one** could be used. Any other frequency will not be orthogonal on the chosen interval L, and it will distort the restored values according to (9) if it is present in the total signal.

Since all sine waves from same set cease to influence each other at the same moment of calculations L, the set can be divided into an arbitrary number of subsets (independent "channels") in an arbitrary way. For example, we can alternate the frequency of channels in turn, group by several sequence numbers, and so on.

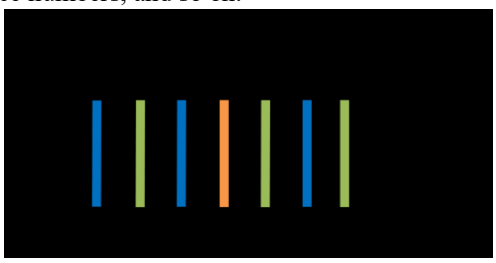

Fig. 5. Orthogonal frequency system channeling example (colors correspond to different channels).

C. Composition and decomposition algorithms

To restore the original information values, it is necessary not only to ensure that in each receiving channel the contributions of all the others are extinguished, but also to consider how this channel is scaled when processing. Scalar product of a function on itself ($\omega^+ = 2\omega$, $\varphi^+ = 2\varphi$, $\omega^- = \varphi^- = 0$):

$$\langle f,f\rangle_L = \frac{1}{2}\int_{t_0}^{t_0+L}[\cos(2\omega t + 2\varphi) + 1]dt = \frac{1}{2\omega}\sin\omega L\cos(\omega L + 2\omega t_0 + 2\varphi) + \frac{L}{2} \quad (15)$$

At the moments of completing the summation ($L = \frac{\pi}{\omega}n, n \in \mathbb{Z}$), it equals to:

$$\langle f,f\rangle_{L=\frac{\pi}{\omega}n} = \frac{L}{2} = \frac{\pi}{2\omega} \quad (16)$$

Thus, to restore the information (linear factors of the basis functions) from the mixed signal $s(t) = Xf(t) + Yg(t)$, we must to multiply corresponding scalar product by $\frac{\pi}{2\omega}$

In summary: for real sine wave systems, choosing the reference frequency step $\Delta\omega$ (or the desired calculation time L for each information numbers packet, that is related by the ratio $\Delta\omega = \frac{2\pi}{L}$) we should form the mixed signal, scaling "odd" or "even" sine series (initial phases on the transmitter and receiver φ_n and ψ_n do not matter) with useful information numbers X_n and Y_n:

$$s_{even}(t) = \sum_{n=0}^{N-1}\left[\pm_n X_n \cos\left(\frac{2\pi}{L}n + \varphi_n\right) \pm_n Y_n \sin\left(\frac{2\pi}{L}n + \varphi_n\right)\right]$$
$$s_{odd}(t) = \sum_{n=0}^{N-1}\left[\pm_n X_n \cos\left(\frac{2\pi}{L}\left(n+\frac{1}{2}\right) + \varphi_n\right) \pm_n Y_n \sin\left(\frac{2\pi}{L}\left(n+\frac{1}{2}\right) + \varphi_n\right)\right] \quad (17)$$

For the next interval L we could use the following scaling numbers. That forms transmitted information flow with data rate limited to time L for one dose.

To restore the information numbers from mixed signal, the receiver must calculate scalar product with each base functions from the series:

$$X_n = \frac{2}{L}\int_{t_0}^{t_0+L} s_{even}(t)\cos\left(\frac{2\pi}{L}n + \psi_n\right)dt = \frac{2}{L}\int_{t_0}^{t_0+L} s_{odd}(t)\cos\left(\frac{2\pi}{L}\left(n-\frac{1}{2}\right) + \psi_n\right)dt$$
$$Y_n = \frac{2}{L}\int_{t_0}^{t_0+L} s_{even}(t)\sin\left(\frac{2\pi}{L}n + \psi_n\right)dt = \frac{2}{L}\int_{t_0}^{t_0+L} s_{odd}(t)\sin\left(\frac{2\pi}{L}\left(n-\frac{1}{2}\right) + \psi_n\right)dt \quad (18)$$

This can be done both with an analog circuit or using digital signal processing (taking into account the necessary sampling intervals and frequency limits with bandwidth filters).

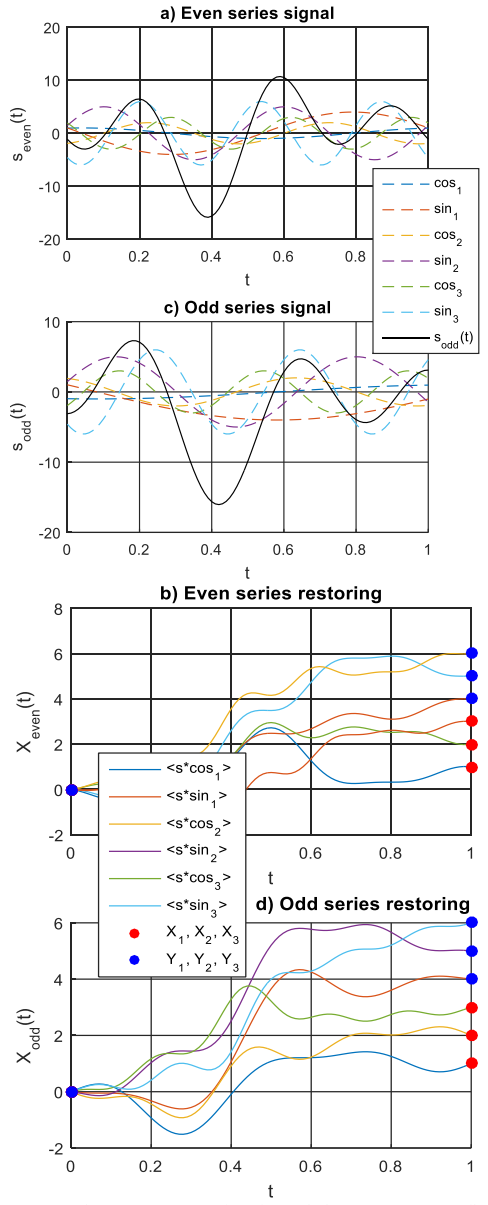

Fig. 6. Signal composition (a, c) and decomposition (b, d) using odd (a, b) and even (c, d) series of orthogonal sine waves, see Matlab code in Appendix A, D.

D. Deviations and errors

The scalar product of base reference signals is strictly zero at a control points only if all relations are strictly precise. If the equality is only approximate, then the numbers recovered at the receiver will differ from the original numbers.

1) Deviation of the initial phases

As was shown in equation (11), for sine waves with different frequencies (base signals) the initial phase does not affect the recovery.

Another situation for the deviations of sine and cosine waves on the same frequency. In this case, the linear error $\frac{L}{2}\cos\varphi^{-}$ will accumulate at each step L (which will cause the numbers to be restored incorrectly), and after a time $\frac{1}{\omega L\cos(\varphi^{-})}$ they will completely diverge. Therefore, it is necessary to synchronize the phase difference of the reference signals at the sender, for example, to get them from a single source of harmonic oscillations and form a phase rotation. Zeroing the current processed value for this at each step L could partially compensate that.

Example of cumulative error is shown in Figure 7 for signals $s_1 = \cos(\pi t)$ and $s_2 = \sin\left(-\pi t - \frac{\pi}{50}\right)$, constructed with Matlab code in Appendix A and E. The same situation but on a large scale is with an arbitrary phase difference φ^{-}.

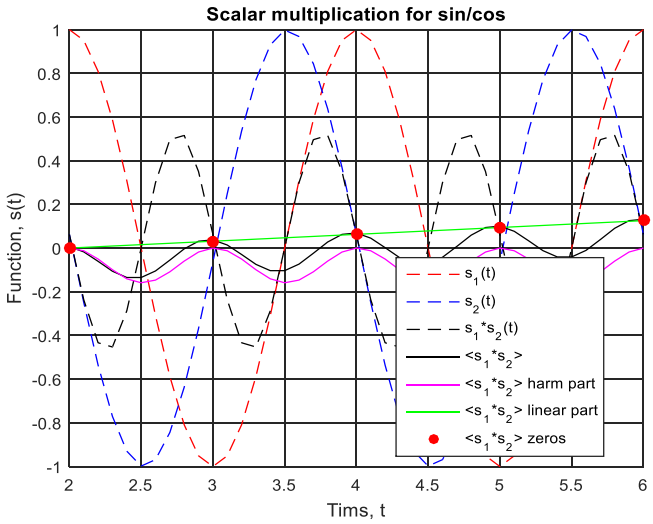

Fig. 7. Restoring error (and shift of zeros) accumulation with almost orthogonal one-frequency sine waves phase deviation $\frac{\pi}{50}$. Linear error is green and periodical is magenta.

2) Frequencies deviation

Using the scalar product formula (9), we can estimate the maximum possible error as $\left|\mathrm{sinc}\frac{\omega^{+}L}{2}\right| + \left|\mathrm{sinc}\frac{\omega^{-}L}{2}\right|$ depending on the presence of non-orthogonal sine wave in a signal for the most unfortunate case of random phases.

For example, with the amplitudes of interference equal to 1 each we obtain the dependence according to Fig. 8 for sine waves with base (smallest) frequency $\omega_0 = \pi$ (see Appendix F):

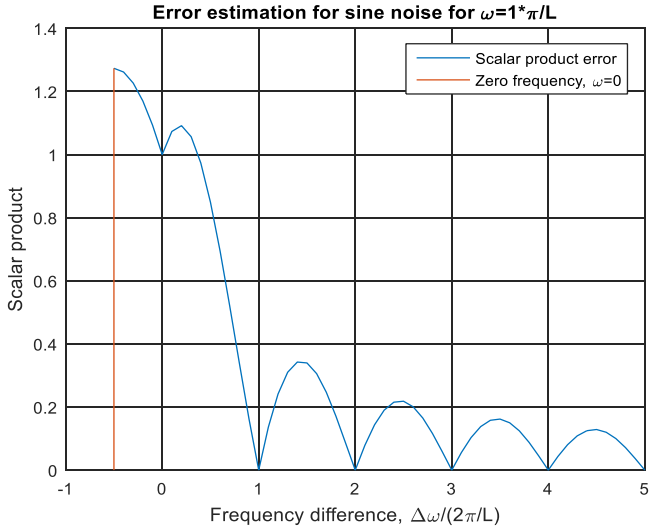

Fig. 8. Restoring maximum error estimation for sine wave shifted from processed. Zeros corresponding to orthogonal frequencies.

3) Deviation of computation time: several computation cycles and guard intervals

If the actual calculation time does not correspond to the planned time L, then in both cases of equal and different

frequencies an error corresponding to equations (6) or (9) will be made when calculating the scalar product. The bias can also occur due to imperfect timers or a shift in the start of the calculation.

To reduce the error (and in the best case, a complete exception), you can use the following tricks:

1. Transmit the summary signal for a time slightly exceeding the required L.
2. Between the changes of the sine waves coefficients (information numbers), insert zero-valued "protective" windows with nothing transmitted (a zero signal will not make any contribution to the calculated).

The value of these corrections could be fixed as the estimated maximum deviations of the calculation time from the generation time or more.

V. Complex sine waves

As shown in Section 4, two arbitrary numbers can be transmitted at each frequency. In particular, we can use this frequency to transmit both components of one complex number, that is one complex number. In other words, at each frequency a real transmission of one complex sine wave is possible.

A. Two orthogonal complex waves

Equation for two complex sine waves f and g, as it was shown in [1]:

$$\begin{cases} f(t) = e^{j(\omega_1 t+\varphi_1)} \\ g(t) = e^{j(\omega_2 t+\varphi_2)} \end{cases} \quad (19)$$

On interval L (starting from time t_0) scalar product for them will be ($\omega^- = \omega_1 - \omega_2, \varphi^- = \varphi_1 - \varphi_2$) as follows:

$$\begin{aligned} \langle f,g\rangle_L &= \int_{t_0}^{t_0+L} \underbrace{e^{j(\omega_1 t+\varphi_1)}}_{f(t)} \underbrace{e^{-j(\omega_1 t+\varphi_1)}}_{g^*(t)} dt \\ &= e^{j\varphi^-} \int_{t_0}^{t_0+L} e^{j\omega^- t} dt \\ &= L \times e^{j(\omega^- t_0+\varphi^-)} \operatorname{csinc}(\omega^- L) \end{aligned} \quad (20)$$

Fig. 9 shows dependence of the current value of the scalar product definition (1) example for the functions $s_1 = e^{j(2\pi t+\pi/2)}$, $s_2 = e^{j(2\pi t+\pi/4)}$ calculated from the initial time t_0 with duration L, realized by MatLab program, see Appendix A and G.

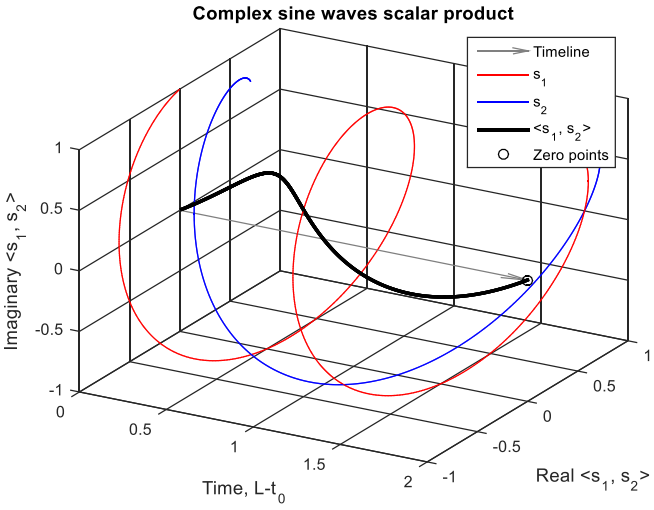

Fig. 9. Scalar product of function for complex sine waves

Here the function $\operatorname{csinc}(x)$ is introduced - the extension of the function $\operatorname{sinc}(x)$ to the space of complex numbers:

$$\begin{aligned} \operatorname{csinc}(x) &= \begin{cases} \dfrac{e^{jx}-1}{jx}, & x \neq 0 \\ 1, & x = 0 \end{cases} \\ &= \begin{cases} \overbrace{\dfrac{\sin x}{x}}^{Re=\operatorname{sinc} x} + j\overbrace{\dfrac{1-\cos x}{x}}^{\mathrm{Im}}, & x \neq 0 \\ 1, & x = 0 \end{cases} \end{aligned} \quad (21)$$

Since $\lim_{x\to 0}(\operatorname{csinc} x) = 1$, this function gives the exact value for the integral, including the case of equal frequencies ($\omega_1 = \omega_2 \leftrightarrow \omega^- = 0$). Particularly, $\langle f,f\rangle_L = \langle g,g\rangle_L = L$. Zero points of scalar product for any different sine waves with $\omega_1 \neq \omega_2$ (that does not depend on initial phases) determine the exact relationship of the difference in their frequencies and the summation time interval:

$$e^{j\omega^- L} = 1 \Leftrightarrow \omega^- = \frac{2\pi}{L} n = \Delta\omega \times n, \qquad n \in \mathbb{Z} \quad (22)$$

B. Deviation of frequencies or phases

What happens if instead of the orthogonal signal $g(t)$ another complex sine wave $\tilde{g}(t)$ will be received? Since the phase $(\omega^- t_0 + \varphi^-)$ is generally be unknown, the scalar product of f and $\tilde{g}$ (with the amplitude a) could be arbitrary, but the deviations of its real and imaginary components will not exceed the error proportional to its module:

$$\begin{aligned} |\Delta Re(\langle f,g\rangle_L)|, |\Delta Im(\langle f,g\rangle_L)| &\le |\operatorname{csinc}(\omega^- L)| \times a \\ &= 2a \frac{1-\cos(\omega^- L)}{(\omega^- L)^2} \end{aligned} \quad (23)$$

shows dependence of the scalar product result (from the initial time $t_0 = 0$ with duration $L = 2$) according to equation (20) for complex sine wave signals $f = f(t) = e^{j\omega_1 t}$ and $g = g(t) = e^{j\omega_2 t}$ (with different frequencies) on the relative frequency difference $\frac{\omega^-}{\Delta\omega} = \frac{\omega_1-\omega_1}{2\pi/L}$ in the Real / Imaginary representation (Fig. 10, а) and Abs / Angle representation (Fig. 10, и). The circles indicate those frequency differences (according (22) correspond to points $\Delta\omega \times n, n \in \mathbb{Z}$) where the scalar product is 0, i.e. the functions are orthogonal in the indicated calculation interval. The graph is built in the MatLab program, see Appendix H.

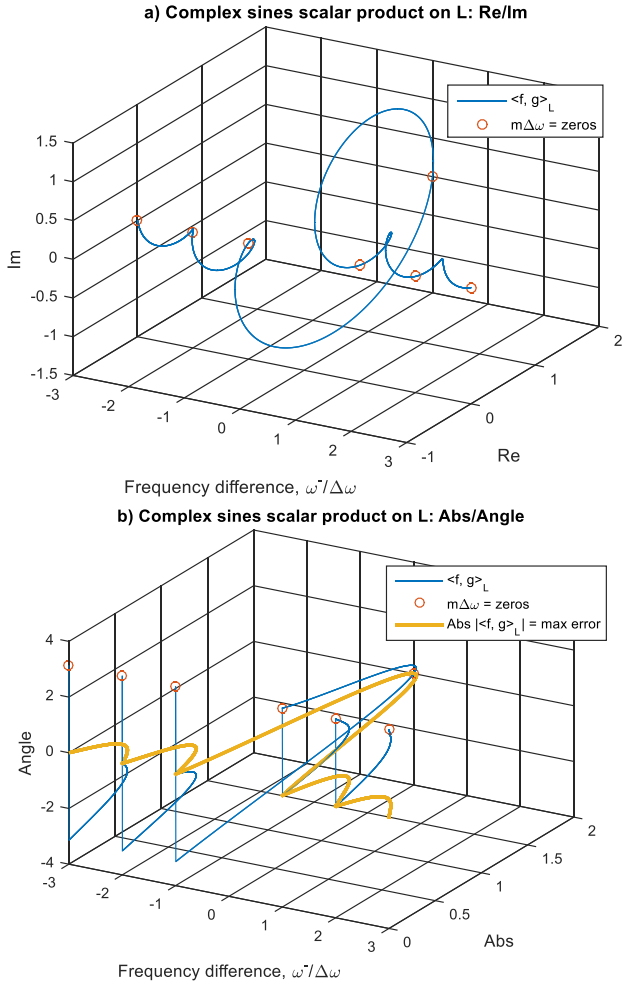

Fig. 10. Scalar product of function for different frequency differences.

It can be seen that certain frequencies (with a step $\Delta\omega = \frac{2\pi}{L}$ between them) do not affect f, since their complex sine wave is orthogonal to it. Others will lead to corresponding error. The same will be true for each of the orthogonal frequencies forming the orthogonal system as follows (but, of course, for them there will be their own distance from the interference frequency $\Delta\omega'$).

This allows us to evaluate both the calculation error associated with the deviation of the frequency at the receiver (from the transmitter frequency) and the influence of extraneous inclusions in the information signal within the bandwidth range.

C. Multiple orthogonal complex sine waves system

If we want to use more than two sine waves for transferring values, then we need **all of them** to be mutually orthogonal at the same known time of the calculation L (any other non-orthogonal components will distort the calculated numbers as shown in the previous subsection) [6].

If the minimum frequency in system is ω_0, and the calculation time is L (so minimal frequency difference is $\Delta\omega = \frac{2\pi}{L}$), then all frequencies that are multiple of the step $\Delta\omega$ it will be orthogonal to it at time L - and only these. But they also will be **orthogonal to each other** at this moment – since they are also separated by a multiple of the same steps $\Delta\omega$. Thus, these two parameters determine any complex sine waves orthogonal system (phases can be arbitrary):

$$\omega_m = \omega_0 + \frac{2\pi}{L}m \Leftrightarrow f_m(t) = e^{j\left(\left(\omega_0+\frac{2\pi}{L}m\right)t+\varphi_m\right)}, \quad m = 0, 1, 2, \ldots \tag{24}$$

Fig. 11 illustrates a system of mutually orthogonal frequencies ω_m for integer non-negative parameters m according to equation (24).

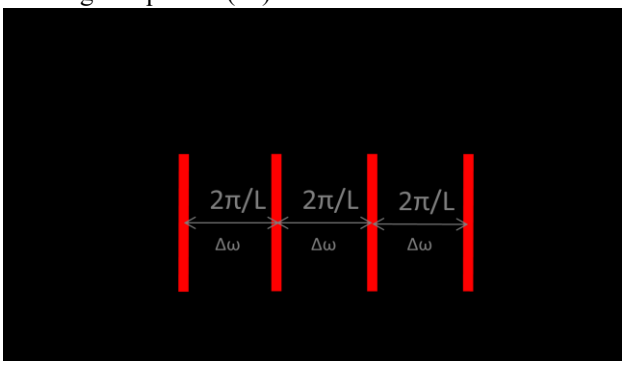

Fig. 11. Orthogonal complex sine waves system visualization.

Thus, the principle of generating a signal $s(t)$ and restoring information (a set of complex amplitude $\{S_m\}$ from that) for M base complex sine waves $f_m(t) = e^{j(\omega_m t+\varphi_m)}, m \in [0..M-1]$, orthogonal on the interval L (i.e. $\omega_m = \omega_0 + \frac{2\pi}{L}m = \omega_0 + \Delta\omega \times m$) is as follows:

$$\begin{aligned} s(t) &= \sum_{m=0}^{M-1} S_m f_m(t) = \sum_{m=0}^{M-1} S_m e^{j(\omega_m t+\varphi_m)} \\ &= e^{j\omega_0 t}\sum_{m=0}^{M-1} S_m e^{j\left(\frac{2\pi}{L}mt+\varphi_m\right)} \\ S_m &= \frac{1}{L}\langle s(t), f_m(t)\rangle_L \\ &= \frac{1}{L}\int_{t_0}^{t_0+L} s(t)e^{-j\left(\left(\omega_0+\frac{2\pi}{L}mt\right)t+\phi_m\right)}dt \end{aligned} \tag{25}$$

D. Decomposition algorithm

Various algorithms for restoring the original numbers using the above formulas could be offered: an analog circuit that directly implements actions, numerical calculation of the integral, or the corresponding matrix form. But the most effective is the algorithm that uses Discrete Fourier Transform (DFT) [3], [7].

The original formula for a signal assembled from basic complex sinusoids is similar to DFT. After some formal transformations for sampling points with a step T (so that the entire time period for calculating the scalar product is $L = NT$, where N where N is the number of samples of the signal):

$$\begin{aligned} \underbrace{s(t)e^{-j\omega_0 t}}_{s'(t)} &= \sum_{m=0}^{M-1}\left(\underbrace{S_m \times e^{j\varphi_m}}_{S_m'}\right)e^{j\frac{2\pi}{L}mt} \Rightarrow s'(nT) \\ &= \sum_{m=0}^{M-1} S_m' e^{j\frac{2\pi}{N}mn} \end{aligned} \tag{26}$$

That is exactly inversed DFT multiplied by N. Therefore, to obtain the S_m' can use the direct discrete Fourier transform (with its fast FFT algorithms [8]) for the samples s'_n, divided by N:

$$S'_m = \frac{1}{N}\sum_{n=0}^{N-1} s'_n e^{-j\frac{2\pi}{N}mn} \quad (27)$$

Important question: what number of samples N should we use? According to the Kotelnikov theorem [3] it should be defined by the doubled maximum signal frequency $2M\Delta\omega$. But this condition arose so that cyclic copies of the continuous spectrum (arising from signal sampling) located in the interval $[-\omega_{max}..\omega_{max}]$ do not overlap each other. In our case there are no negative frequencies in spectrum (all difference frequencies are $\Delta\omega$ multiplied by positive integer). Therefore, the minimum time sampling step is **two times less** and is equal to the inverse maximum frequency:

$$T \le \frac{1}{f_{max}} = \frac{2\pi}{\omega_{max}} = \frac{2\pi}{M\Delta\omega} \quad (28)$$

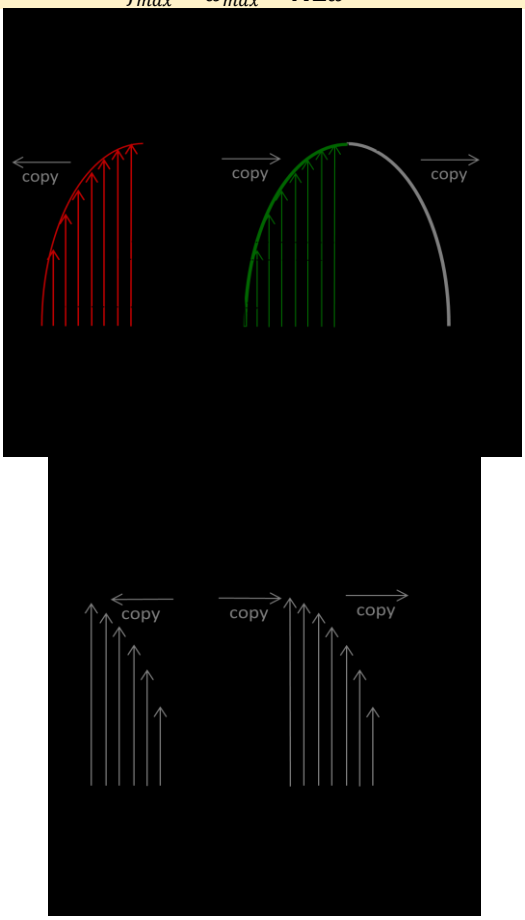

Fig. 12. Common sampled signal spectrum (a) and case of orthogonal sine waves system (b)

Fig. 12 illustrates the difference between the sampled spectrum of some arbitrary signal (a) containing negative frequencies and a set of complex sine waves with only positive frequency differences (b). In both cases, the spectrum duplicated with copies, but the minimum frequency distance of copying without overlapping for case (b) can be 2 times smaller.

The required number of signal samples in this case is $N \ge M$. Only M samples are sufficient, even for very high frequencies of the mixed signal. The Fast Fourier Transform (FFT) algorithm requires $\sim M \log M$ operations. Hence the initial coefficients (information numbers):

$$\mathrm{S_m} = S'_m e^{-j\varphi_m} = \frac{e^{-j\varphi_m}}{M}\sum_{n=0}^{M-1}\left(s_n e^{e^{-j\omega_0 nT}}\right)e^{-j\frac{2\pi}{M}mn} \quad (29)$$

Fig. 13 (a) shows the signal $s(t)$, which is the sum of 4 complex sine waves (base frequency $\omega_0 = \pi$; the step between the frequencies $\Delta\omega = \pi/8$, the initial phases are random), scaled by random complex numbers S_m (which are useful information transmitted) and signal samples s_n needed to restore these numbers. Fig. 13 (b) shows the result S_m of applying the discrete Fourier transform (exactly coinciding with the original information numbers) to these samples set $\{s_n\}$ and the envelope of the discrete spectrum.

Figures obtained in MatLab program – see Appendix I.

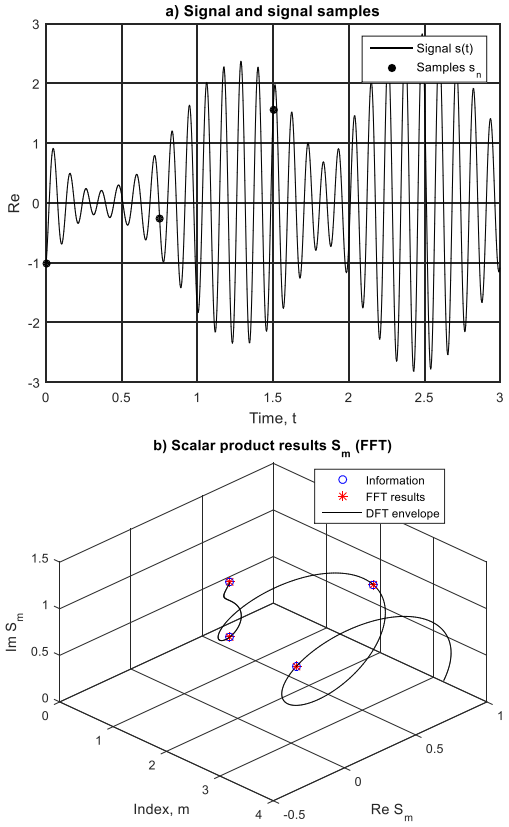

Fig. 13. Signal as the set of orthogonal complex sine waves (a) and information (scale factors) restoring using FFT (b).

The processing algorithm is as follows: getting M samples s_n from the original signal $s(t)$, multiplying each of them by $e^{-j\omega_0 nT}$. Carrying out FFT for the obtained set of numbers, obtaining S'_m. To obtain the initial data S_m – scaling with $\times\frac{1}{M}$ and phase rotation according to initial complex sine waves: $\times e^{-j\varphi_m}$. These phases can be determined, for example, by transmitting known set of numbers at all base frequencies, for example {1,1,1 ... 1}.

VI. Conclusion

The article briefly described the issues of information transfer (complex numbers) using a system of orthogonal functions. The design of such a system is discussed (and possible errors with deviations) as well as the MatLab algorithm is proposed to recover information from the generated mixed signal with operation time slowly increases from the number of transmitted numbers (even for high frequencies). Thus, this method can be an effective way of transmitting information.

APPENDIX

A. Matlab code 1. Numerical integral as function of upper limit function

integralfuncrect.m

```
%% Numerical integral as a function
% dt - integration step, tn = t0+dn*n
%  f   - function vector = [f1(t0) f1(t1) ...;
f2(t0) f2(t1) ...; ...]
function i = integralfuncrect(dt, f)
    v = zeros(1, size(f, 1));            % current
integral value
    i = zeros(size(f, 2), size(f, 1));  % function
of integral, I(0) = 0;
    for n = 2:size(f, 2);
        v = v + conj(f(:,n))';
        i = [i; v];
    end
    i = i*dt;
end
```

B. Matlab code 2. Two sine waves scalar product

```
w1 = pi;                          % 1st frequency
p1 = 0;                           % 1st phase
s1 = @(t) cos(w1*t+p1);           % 1st sine wave

w2 = 2/3*pi;                      % 2nd frequency
p2 = pi/2;                        % 2nd phase
s2 = @(t) cos(w2*t+p2);           % 2nd sine wave

dt = 0.01;                        % time step


L = 2*pi/abs(w1-w2);              % zero times
time = 0:dt:3.5*L;                % timeline
izero = round((0:3)/dt*L)+1;      % zero indexes

% scalar function in time
Integral = integralfuncrect(dt,
[s1(time).*s1(time); s1(time).*s2(time);
s2(time).*s2(time)]);

%% Plot

figure();
hold on; grid on;

title('<s1, s2>(t)');
xlabel('Time, t');
ylabel('Value f(t)');

plot(time,                          s1(time),'r:');
% plot s1
plot(time,                          s2(time),'b:');
% plot s2
plot(time,              s1(time).*s2(time),'k--');
% plot s2

plot(time,      integral(:,      1),      'r-');
% plot <s1, s1>
plot(time,      integral(:,      3),      'b-');
% plot <s1, s2>
plot(time,      integral(:,      2),      'k-');
% plot <s1, s2>

scatter(izero*dt,    integral(izero,    1),    50,
'filled', 'r');
scatter(izero*dt, integral(izero, 3), 50, 'b');
scatter(izero*dt,    integral(izero,    2),    50,
'filled', 'k');

legend('s1(t)',   's2(t)',   's1(t)*s2(t)',   '<s1,
s1>', '<s2, s2>', '<s1, s2>');
```

C. Matlab code 3. Two sine waves scalar product plane

```
%% sinc(x)cos(x+u)+sinc(y)cos(y+u) plane

N = 3;              % number of [-pi pi] intervals
shown
acc = 20;           % accuracy of a plot

u = 2*pi*rand(1,1); % random u
v = 2*pi*rand(1,1); % random v

%% Processing
dx = pi/acc;
dy = pi/acc;

[x, y] = meshgrid(-pi*N:dx:pi*N, -pi*N:dy:pi*N);
% xy matrix
z = (sinc(x/pi).*cos(x+u) + sinc(y/pi).*cos(y+v));
% sinc = sin(pi*x)/(pi*x)

scale = max(max(abs(z)))*3;         % peak height
calculating
z(abs(z)<0.00001) = scale;          % peak instead
zero

%% Plotting
hold on; grid on; view(143, 48);

title('Peaks on zeros of
<f,g>=sinc(x)cos(x+u)+sinc(y)cos(y+v)');
xlabel('x'); ylabel('y'); zlabel('<f, g> peaked');

surf(x, y, z, 'EdgeColor', 'none');
```

D. Matlab code 4. Orthogonal sines based signal composition and restoring

```
w0 = pi;         % base frequency
t0 = 0;          % start time
N = 1;           % cycles
acc = 100;

X = [1 2 3];  % information 1
Y = [4 5 6];  % information 2

phs = 2*pi*rand(size(X, 2), 1)-pi;  % random
phases for cos

%% Processing
L = pi/w0;
dt = L/acc;
t1 = t0+L*N;
time = t0:dt:t1;           % timeline
tzero = t0:L:t1;           % zero points


sn_even_cos = @(t, n) -cos(2*w0*n*t+phs(n));
% even (Fourier) cos
sn_even_sin = @(t, n) sin(2*w0*n*t+phs(n));
% even (Fourier) sin

sn_odd_cos = @(t, n) cos(2*w0*(n-1/2)*t+phs(n));
% odd series cos
sn_odd_sin = @(t, n) sin(2*w0*(n-1/2)*t+phs(n));
% odd series sin

%% Form even series (Fourier)
subplot(2, 2, 1);
hold on; grid on;

title('a) Even series signal');
xlabel('t'); ylabel('s {even}(t)');

s_even = zeros(1, size(time, 2));        % init
signal 1
for i=1:size(X, 2)
```

```
    plot(time, X(i)*sn_even_cos(time, i), '--',
'DisplayName', strcat('cos_',num2str(i)));
    plot(time, Y(i)*sn_even_sin(time, i), '--',
'DisplayName', strcat('sin_',num2str(i)));

    s_even = s_even + X(i)*sn_even_cos(time, i) +
Y(i)*sn_even_sin(time, i);
end

plot(time, s_even, 'k-', 'DisplayName',
's_{even}(t)');

legend('Location', 'best');

%% Form odd series
subplot(2, 2, 3);
hold on; grid on;

title('c) Odd series signal');
xlabel('t'); ylabel('s_{odd}(t)');

s_odd = zeros(1, size(time, 2));        % init
signal 1
for i=1:size(X, 2)
    plot(time, X(i)*sn_odd_cos(time, i), '--',
'DisplayName', strcat('cos ',num2str(i)));
    plot(time, Y(i)*sn_odd_sin(time, i), '--',
'DisplayName', strcat('sin_',num2str(i)));

    s_odd = s_odd + X(i)*sn_odd_cos(time, i) +
Y(i)*sn_odd_sin(time, i);
end

plot(time, s_odd, 'k-', 'DisplayName',
's_{odd}(t)');

legend('Location', 'best');

%% Processing: new phases
phs = 2*pi*rand(size(X, 2), 1)-pi;  % random
phases for cos

%% Restoring even series (Fourier)
subplot(2, 2, 2);
hold on; grid on;

title('b) Even series restoring');
xlabel('t'); ylabel('X_{even}(t)');

for i=1:size(X, 2)
    integral = 2/L*integralfuncrect(dt,
[s_even.*sn_even_cos(time, i);
s_even.*sn_even_sin(time, i)]);

% plot <s, cos>
    plot(time, integral(:, 1), 'DisplayName',
strcat('<s*cos_',num2str(i),'>'));
% plot <s, sin>
    plot(time, integral(:, 2), 'DisplayName',
strcat('<s*sin_',num2str(i),'>'));

    scatter(tzero, integral(1+tzero/dt, 1), 50,
'filled', 'r', 'DisplayName',
strcat('X ',num2str(i))); % <s1, s1> @ zeros
    scatter(tzero, integral(1+tzero/dt, 2), 50,
'filled', 'b', 'DisplayName',
strcat('Y_',num2str(i))); % <s1, s1> @ zeros
end

legend('Location', 'best');

%% Restoring odd series
subplot(2, 2, 4);
hold on; grid on;

title('d) Odd series restoring');
xlabel('t'); ylabel('X_{odd}(t)');

for i=1:size(X, 2)
    integral = 2/L*integralfuncrect(dt,
[s_odd.*sn_odd_cos(time, i);
s_odd.*sn_odd_sin(time, i)]);

% plot <s1, s1>
    plot(time, integral(:, 1), 'DisplayName',
strcat('<s*cos_',num2str(i),'>'));
% plot <s1, s1>
    plot(time, integral(:, 2), 'DisplayName',
strcat('<s*sin_',num2str(i),'>'));

    scatter(tzero, integral(1+tzero/dt, 1), 50,
'filled', 'r', 'DisplayName',
strcat('X_',num2str(i))); % <s1, s1> @ zeros
    scatter(tzero, integral(1+tzero/dt, 2), 50,
'filled', 'b', 'DisplayName',
strcat('Y_',num2str(i))); % <s1, s1> @ zeros
end

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

E. Matlab code 5. Phase error summation

```
%% Parameters
L = 1;          % time to sum
N = 4;          % number of cycles

t0 = 2;         % start time
dt = L/10;      % time step

w = pi/L;       % sin/cos frequency

ph1 = 0;          % s1 (cos) phase
ph2 = pi/2-pi/50; % s2 (sin) phase

%% Processing
time = t0:dt:(t0+N*L);  % timeline
tzero = t0:L:(t0+N*L);  % zero points

ph_dif = ph1 - ph2;     % phases dif
ph_sum = ph1 + ph2;     % phases sum

s = @(t, w, ph) cos(w*t+ph);    % signal function

integral = integralfuncrect(dt, s(time, w,
ph1).*s(time, w, ph2));    % scalar dot func
i2 = @(t) 1/2/w*cos(w*t+2*w*t0+ph_sum).*sin(w*t);
% harmonical part
i3 = @(t) t/2*cos(ph_dif);
% linear part

%% Plotting
figure(); hold on; grid on;
title('Scalar multiplication for sin/cos');
xlabel('Tims, t');
ylabel('Function, s(t)');

plot(time, s(time, w, ph1),'r--', 'DisplayName',
's_1(t)');     % plot s1
plot(time, s(time, w, ph2),'b--', 'DisplayName',
's_2(t)');     % plot s2
plot(time, s(time, w, ph1).*s(time, w, ph2),'k--',
'DisplayName', 's_1*s_2(t)');   % plot s1*s2

plot(time, integral(:, 1), 'k', 'DisplayName',
'<s_1*s_2>');       % plot <s1, s2>
plot(time, i2(time)-i2(t0), 'm', 'DisplayName',
'<s_1*s_2> harm part');   % harm <s1, s2>
plot(time, i3(time)-i3(t0), 'g', 'DisplayName',
'<s_1*s_2> linear part'); % linear <s1, s2>

scatter(tzero, i2(tzero)+i3(tzero)-i2(t0)-i3(t0),
50, 'filled', 'r', 'DisplayName', '<s_1*s_2>
zeros'); % <s1, s2> @ zeros

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

F. Matlab code 6. Error estimation for sine noise

```
L = 1;                  % summation time
r = 1;                  % series base: 1=odd,
2=even, ...
N = 5;                  % steps number (to each
```

```
direction)

w0 = pi/L*r;            % ortho base
w_step = 2*pi/L;        % ortho step

dw_rel = -r/2:0.1:N;    % freq shift in steps,
from zero frequency
w2 = w0 + dw_rel*w_step;% noise frequency

w_sum = w0 + w2;        % summary freq
w_dif = w0 - w2;        % differential freq

sinc2 = @(x) sinc(x/pi);% instead of matlab sinc =
sin(pi*x)/(pi*x)

%% Plotting
hold on; grid on;

title(strcat('Error estimation for sine noise for
\omega=',num2str(r),'*\pi/L'));
xlabel('Frequency difference,
\Delta\omega/(2\pi/L)');
ylabel('Scalar product');

plot(dw_rel, abs(sinc2(w_sum*L/2)) +
abs(sinc2(w_dif*L/2)));   % |sinc(w+/2L)| +
|sinc(w-/2L)|;
plot([-r/2 -r/2],[0 2*abs(sinc2(w0*L/2))]);

legend('Scalar product error', 'Zero frequency,
\omega=0');
```

G. Matlab code 7. Two complex sine waves scalar product

```
s = [1 2*pi pi/2; 1 pi  pi/4];              %
signal params [A w phase]
N = 1000;                                   %
accuracy = samples number
C = 1;                                      %
periods to plot
%% Processing
csine  = @(s, t) s(1)*exp(j*(s(2)*t+s(3))); % sine
wave: A exp(j(wt+phi));

L = C*2*pi/abs(s(1,2)-s(2,2));              % C
periods with duration L
dt = L/(N-1);                               % time
step
t = dt*(0:N-1);                             %
timeline = 2 periods
sp = integralfuncrect(dt, csine(s(1,:),
t).*conj(csine(s(2,:), t))); % scalar product

%% Display
plotc = @(t, x, varargin) plot3(t, real(x),
imag(x), varargin{:}); % plot as (t,Re,Im)

figure(); hold on; grid on; view(30, 30);
title('Complex sine waves scalar product');
xlabel('Time'); ylabel('Real');
zlabel('Imaginary');
quiver3(0, 0, 0, L, 0, 0, 0, 'Color', [.5 .5 .5]);
% Time axe

plotc(t, csine(s(1,:), t), 'r--');
% display s1
plotc(t, csine(s(2,:), t), 'b-');
% display s2
plotc(t, sp, 'k', 'LineWidth', 2);
% display <s1, s2>

tmp = max(abs(sp))-abs(sp);
% display local mininums
[pks, locs] = findpeaks(tmp);
scatter3(t(locs), real(sp(locs)), imag(sp(locs)),
'k');

legend('Timeline', 's_1', 's_2', '<s_1, s_2>',
'Zero points');
```

H. Matlab code 8. Scalar product for frequency difference

```
t0 = 0;             % start time
L = 2;              % duration (summation time)
dp = 0;             % phase difference
N = 1000;           % accuracy to plot
%% Processing
W = 2*pi/L;         % ortho system frequency step
(for L)
dw = W/N;           % plotting step

w = -3*W:dw:3*W;    % frequencies line
wz = -3*W:W:3*W;    % zero frequencies

err = @(w) L*exp(j*dp)*exp(j*w*t0).*csinc(w*L); %
scalar product function
e = err(w);         % error value
z = err(wz);        % zeros

%% Display in Re, Im
figure(); hold on; grid on; view(30, 30);
title('a) Complex sines scalar product on L:
Re/Im');
xlabel('Frequency difference, \omega^-
/\Delta\omega'); ylabel('Re'); zlabel('Im');

plot3(w/W, real(e), imag(e));      % plot scalar
product
scatter3(wz/W, real(z), imag(z));  % plot zeros
legend('<f, g>_L', 'm\Delta\omega = zeros');

%% Display in Re, Im
figure(); hold on; grid on; view(30, 30);
title('b) Complex sines scalar product on L:
Abs/Angle');
xlabel('Frequency difference, \omega^-
/\Delta\omega'); ylabel('Abs'); zlabel('Angle');

plot3(w/W, abs(e), angle(e));   % plot scalar
product
scatter3(wz/W, abs(z), angle(z)); % plot zeros
plot(w/W, L*power(2*(1-cos(w*L))./power(w*L, 2),
1/2), 'LineWidth', 2); % plot abs
legend('<f, g>_L', 'm\Delta\omega = zeros', 'Abs
|<f, g>_L| = max error');
```

csinc.m

```
function y = csinc(x)
    iz = find(x==0);
    y = -j*(exp(j*x)-1)./x;
    y(iz) = 1;
end
```

I. Matlab code 9. FFT restore algorithm

```
%% Parameters
L = 3;                          % time to detect
w0 = 50*pi/L;                   % ortho system
base frequency
M = 4;                          % number of waves
(information numbers)
Z = rand(1, M)+j*rand(1, M);    % information
numbers
env_scale = 128;                % DFT envelope
scale factor

%% Preprocessing
csine = @(t, w, ph) exp(j*(w*t+ph)); % wave signal

w_step = 2*pi/L;                    % system
frequency step
ws = w0 + w_step*(0:size(Z, 2)-1);  % system
frequencies vector
ps_sender = 1*2*pi*rand(1, M);      % system
phases (random) vector for sender
ps_receiver = ps_sender;            % system
phases (random) vector for receiver

N = M;                              % number of
```

```
samples: N>=M
dt = L/N;                              % sampling
time step
t = dt*(0:N-1);                        % sampling
timeline
tick = dt/256;                         % continuous
time step (for plot)
time = tick*(0:L/tick-1);              % continuous
timeline

%% Ortho signal forming
s = 0*time;
samples = 0*t;
for n = 1:M
    s = s + Z(n)*csine(time, ws(n), ps_sender(n));
    samples = samples + Z(n)*csine(t, ws(n),
ps_sender(n));
end

%% FFT restore
Z_fft = 1/N*fft(samples.*exp(-j*(t*w0)));
% rotate with t = {n*dt}, n=0..N-1;
Z_fft(1:M) = Z_fft(1:M).*exp(-j*ps_sender);
% rotate phases back
Z_env = dft_scale(Z_fft, env_scale);% DFT envelope

%% Display signal and samples
figure(); hold on; grid on;
title('a) Signal'); xlabel('t'); ylabel('Re');
zlabel('Im');

plot3(time, real(s), imag(s), 'k');
% display continuous
scatter3(t, real(samples), imag(samples), 'k',
'filled');   % display signal samples

%% Display numbers & FFT results
figure(); hold on; grid on; view(45, 45);
title('d) Scalar product results(FFT and
matrix)');
xlabel('m'); ylabel('Re S_m'); zlabel('Im S_m');

scatter3(0:N-1, real(Z), imag(Z), 'k', 'filled',
'DisplayName', 'FFT results');
scatter3(0:N-1, real(Z_fft), imag(Z_fft), 'r',
'DisplayName', 'FFT results');
plot3((0:env_scale*N-1)/env_scale, real(Z_env),
imag(Z_env), '-', 'DisplayName', 'DFT envelope');
legend('Location', 'best');
```

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] E. Tikhonov, M. Sneps-Sneppe. “Introduction to signal processing: sampled signals”, International Journal of Open Information Technologies, Vol. 7, No 7, June 2019, ISSN 2307-8162.

[4] R.N. Bracewell. “The Fourier Transform”. Journal “V mire nauki”, (Scientific American. Russian Language edition. Articles of Scientific American, June 1989, Vol. 260, No. 6), No 8, 1989, pages 48–56.“Mir”, Moscow, ISSN 0208-0621.

[5] S. Lipschutz; M. Lipson (2009). Linear Algebra (Schaum’s Outlines) (4th ed.). McGraw Hill. ISBN 978-0-07-154352-1.

[6] R.R. Gallager. "Information theory and reliable communication" (1968), New York: John Wiley & Sons, Inc., ISBN W-471-29048-3.

[7] Emmanuel C. Ifeachor, Barrie W. Jervis (2001) "Digital Signal Processing. A practical approach. Second edition". Prentice Hall; ISBN: 0201-59619-9.

[8] [8] A.B Sergienko, Digital signal procrssing (Tsifrovaja obrabotka signalov, in Russian) // Piter, 2002, ISBN 5-318-00666-3.