Computational and Applied Mathematics (2020) 39:329
https://doi.org/10.1007/s40314-020-01373-7

Discrete octonion Fourier transform and the analysis of discrete 3-D data

Łukasz Błaszczyk[1]

Received: 20 December 2019 / Revised: 4 September 2020 / Accepted: 27 October 2020 /
Published online: 22 November 2020
© The Author(s) 2020

Abstract
The purpose of this article is to further develop the theory of octonion Fourier transformations (OFT), but from a different perspective than before. It follows the earlier work by Błaszczyk and Snopek, where they proved a few essential properties of the OFT of real-valued functions of three continuous variables. The research described in this article applies to discrete transformations, i.e. discrete-space octonion Fourier transform (DSOFT) and discrete octonion Fourier transform (DOFT). The described results combine the theory of Fourier transform with the analysis of solutions for difference equations, using for this purpose previous research on algebra of quadruple-complex numbers. This hypercomplex generalization of the discrete Fourier transformation provides an excellent tool for the analysis of 3-D discrete linear time-invariant (LTI) systems and 3-D discrete data.

Keywords Hypercomplex numbers · Octonion Fourier transform · Discrete-time systems · Fast Fourier transform

Mathematics Subject Classification 30G35 (primary) · 65T50 · 94A12

1 Introduction

Research in the field of signal and system processing is not only an analysis of the functions of a continuous variable (interpreted as time or space) and their representations in the frequency domain, but also in the field of discrete variables, which we naturally obtain as a result of the process of digitization (i.e. in connection with sampling of signals). As in the case of classical signal processing, so the discrete counterpart of this theory has so far mainly focused on signals with real and complex values, as well as their complex spectra. In recent years,

Communicated by Apala Majumdar.

The research was supported by WUT grant No. 504/04239/1120.

✉ Łukasz Błaszczyk
l.blaszczyk@mini.pw.edu.pl

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warsaw, Poland

$\mathbf{e}_i \cdot \mathbf{e}_j$		$\mathbf{e}_j$							
		1	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_6$	$\mathbf{e}_7$
$\mathbf{e}_i$	1	1	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_6$	$\mathbf{e}_7$
	$\mathbf{e}_1$	$\mathbf{e}_1$	-1	$\mathbf{e}_3$	$-\mathbf{e}_2$	$\mathbf{e}_5$	$-\mathbf{e}_4$	$-\mathbf{e}_7$	$\mathbf{e}_6$
	$\mathbf{e}_2$	$\mathbf{e}_2$	$-\mathbf{e}_3$	-1	$\mathbf{e}_1$	$\mathbf{e}_6$	$\mathbf{e}_7$	$-\mathbf{e}_4$	$-\mathbf{e}_5$
	$\mathbf{e}_3$	$\mathbf{e}_3$	$\mathbf{e}_2$	$-\mathbf{e}_1$	-1	$\mathbf{e}_7$	$-\mathbf{e}_6$	$\mathbf{e}_5$	$-\mathbf{e}_4$
	$\mathbf{e}_4$	$\mathbf{e}_4$	$-\mathbf{e}_5$	$-\mathbf{e}_6$	$-\mathbf{e}_7$	-1	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$
	$\mathbf{e}_5$	$\mathbf{e}_5$	$\mathbf{e}_4$	$-\mathbf{e}_7$	$\mathbf{e}_6$	$-\mathbf{e}_1$	-1	$-\mathbf{e}_3$	$\mathbf{e}_2$
	$\mathbf{e}_6$	$\mathbf{e}_6$	$\mathbf{e}_7$	$\mathbf{e}_4$	$-\mathbf{e}_5$	$-\mathbf{e}_2$	$\mathbf{e}_3$	-1	$-\mathbf{e}_1$
	$\mathbf{e}_7$	$\mathbf{e}_7$	$-\mathbf{e}_6$	$\mathbf{e}_5$	$\mathbf{e}_4$	$-\mathbf{e}_3$	$-\mathbf{e}_2$	$\mathbf{e}_1$	-1

Fig. 1 Multiplication rules in octonion algebra

however, more and more works have started to appear, which authors use in their research hypercomplex algebras, among others quaternions and octonions (Brackx et al. 2013; Hahn and Snopek 2016; Lian 2019; Snopek 2015; Wang et al. 2017). The area of applications is focused so far on the study of neural networks (Popa 2016, 2018), analysis of color and multispectral images (Ell et a. 2014; Gao and Lam 2014; Gomes et al. 2017; Grigoryan and Agaian 2018; Lazendić et al. 2018a, b; Sheng et al. 2018), as well as the biomedical signals processing (Delsuc 1988; Klco et al. 2017).

Octonions are defined as the 8-tuple of real numbers, i.e.

$$o = o_0 + o_1\mathbf{e}_1 + o_2\mathbf{e}_2 + o_3\mathbf{e}_3 + o_4\mathbf{e}_4 + o_5\mathbf{e}_5 + o_6\mathbf{e}_6 + o_7\mathbf{e}_7 \in \mathbb{O}, \quad o_0, \ldots, o_7 \in \mathbb{R},$$

where $\mathbf{e}_1, \ldots, \mathbf{e}_7$ are seven different octonion imaginary units, each satisfying the property $\mathbf{e}_i^2 = \mathbf{e}_i \cdot \mathbf{e}_i = -1$, $i = 1, \ldots, 7$, and other octonion multiplication rules shown in Fig. 1. Octonions form a non-associative and a non-commutative algebra (Baez 2002).

In previous studies, we focused on discussing the properties of the octonion Fourier transform (OFT) of real-valued functions of three variables (and in the case of some properties, we extended this to octonion-valued functions) (Błaszczyk 2018, 2019, 2020; Błaszczyk and Snopek 2017). OFT is given by a formula

$$U_{\mathrm{OFT}}(f_1, f_2, f_3) = \int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}} u(x_1, x_2, x_3) \cdot \mathrm{e}^{-2\pi\mathbf{e}_1 f_1 x_1} \cdot \mathrm{e}^{-2\pi\mathbf{e}_2 f_2 x_2} \cdot \mathrm{e}^{-2\pi\mathbf{e}_4 f_3 x_3}\, \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}x_3, \tag{1}$$

where $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_4$ are three of the seven octonion imaginary units and the octonion exponential function is defined by the power series (Błaszczyk 2020), similarly as for the complex numbers and quaternions. In previous investigations (Błaszczyk 2020; Błaszczyk and Snopek 2017) it has been shown that (1) is well defined (i.e. the inverse transform theorem is valid) and some properties of the OFT, analogous to the properties of the complex and quaternion Fourier transform (e.g. Hermitian symmetry, shift theorem, Plancherel-Parseval theorems and derivative theorem) were also derived.

$\mathbf{E}_i \odot \mathbf{E}_j$		$\mathbf{E}_j$							
		1	$\mathbf{E}_1$	$\mathbf{E}_2$	$\mathbf{E}_3$	$\mathbf{E}_4$	$\mathbf{E}_5$	$\mathbf{E}_6$	$\mathbf{E}_7$
$\mathbf{E}_i$	1	1	$\mathbf{E}_1$	$\mathbf{E}_2$	$\mathbf{E}_3$	$\mathbf{E}_4$	$\mathbf{E}_5$	$\mathbf{E}_6$	$\mathbf{E}_7$
	$\mathbf{E}_1$	$\mathbf{E}_1$	-1	$\mathbf{E}_3$	$-\mathbf{E}_2$	$\mathbf{E}_5$	$-\mathbf{E}_4$	$\mathbf{E}_7$	$-\mathbf{E}_6$
	$\mathbf{E}_2$	$\mathbf{E}_2$	$\mathbf{E}_3$	-1	$-\mathbf{E}_1$	$\mathbf{E}_6$	$\mathbf{E}_7$	$-\mathbf{E}_4$	$-\mathbf{E}_5$
	$\mathbf{E}_3$	$\mathbf{E}_3$	$-\mathbf{E}_2$	$-\mathbf{E}_1$	1	$\mathbf{E}_7$	$-\mathbf{E}_6$	$-\mathbf{E}_5$	$\mathbf{E}_4$
	$\mathbf{E}_4$	$\mathbf{E}_4$	$\mathbf{E}_5$	$\mathbf{E}_6$	$\mathbf{E}_7$	-1	$-\mathbf{E}_1$	$-\mathbf{E}_2$	$-\mathbf{E}_3$
	$\mathbf{E}_5$	$\mathbf{E}_5$	$-\mathbf{E}_4$	$\mathbf{E}_7$	$-\mathbf{E}_6$	$-\mathbf{E}_1$	1	$-\mathbf{E}_3$	$\mathbf{E}_2$
	$\mathbf{E}_6$	$\mathbf{E}_6$	$\mathbf{E}_7$	$-\mathbf{E}_4$	$-\mathbf{E}_5$	$-\mathbf{E}_2$	$-\mathbf{E}_3$	1	$\mathbf{E}_1$
	$\mathbf{E}_7$	$\mathbf{E}_7$	$-\mathbf{E}_6$	$-\mathbf{E}_5$	$\mathbf{E}_4$	$-\mathbf{E}_3$	$\mathbf{E}_2$	$\mathbf{E}_1$	-1

Fig. 2 Multiplication rules in $\mathbb{F}$

A major difficulty in these works was the lack of significant properties of octonion multiplication, i.e. commutativity and associativity. Non-commutativity is also encountered with Fourier transforms based on quaternion algebra and (in general) Clifford algebras (Brackx et al. 2013). In order to deal with these problems, we have defined the algebra of quadruple-complex numbers (based on the double-complex numbers introduced by Ell (1993); Kurman (1958)). Like octonions, we define the algebra of order 8 over the field of real numbers and each element of this algebra will be identified with the 8-tuple of real numbers (Błaszczyk 2019, 2020), i.e.

$$p = p_0 + p_1\mathbf{E}_1 + p_2\mathbf{E}_2 + p_3\mathbf{E}_3 + p_4\mathbf{E}_4 + p_5\mathbf{E}_5 + p_6\mathbf{E}_6 + p_7\mathbf{E}_7 \in \mathbb{F}, \quad p_0, \ldots, p_7 \in \mathbb{R}.$$

Addition in $\mathbb{F}$ is defined in a classical way—element-wise, and multiplication (denoted as $\odot$) rules are shown in Fig. 2. We can see that imaginary units in $\mathbb{F}$ follow the analogous (although significantly different) rules as the octonion multiplication, i.e.

$$\mathbf{E}_1 \odot \mathbf{E}_1 = \mathbf{E}_2 \odot \mathbf{E}_2 = -\mathbf{E}_3 \odot \mathbf{E}_3 = \mathbf{E}_4 \odot \mathbf{E}_4 = -\mathbf{E}_5 \odot \mathbf{E}_5 = -\mathbf{E}_6 \odot \mathbf{E}_6 = \mathbf{E}_7 \odot \mathbf{E}_7 = -1.$$

It is worth pointing out that, just as in the case of octonions, algebra $\mathbb{F}$ is generated by three imaginary units, i.e. $\mathbf{E}_1$, $\mathbf{E}_2$ and $\mathbf{E}_4$. Each other imaginary unit can be obtained by multiplying these three units, i.e. $\mathbf{E}_3 = \mathbf{E}_1 \odot \mathbf{E}_2$, $\mathbf{E}_5 = \mathbf{E}_1 \odot \mathbf{E}_4$, $\mathbf{E}_6 = \mathbf{E}_2 \odot \mathbf{E}_4$ and $\mathbf{E}_7 = \mathbf{E}_1 \odot \mathbf{E}_2 \odot \mathbf{E}_4$. However, unlike octonions, the three basic imaginary units of algebra $\mathbb{F}$ are commutative. The multiplication in $\mathbb{F}$ is commutative and associative, however not every non-zero element of $\mathbb{F}$ has its inverse, which is a property common to Clifford algebras. This algebra has already appeared in the literature, in particular in Felsberg et al. (2001) (where it was denoted as hypercomplex algebra $\mathcal{H}_4$). By using operations in the quadruple algebra the notation of formulas for determining many OFT properties could be reduced to simple equations known from classical theory. We discussed that in detail in Błaszczyk (2019, 2020).

From the point of view of numerical calculations and practical applications, it is also important to develop discrete equivalents of the transformations under consideration and their properties. In the case of the classical Fourier transformation, this subject is well known, as in the case of quaternions (Bahri and Surahman 2013; Ell et a. 2014; Sangwine and Bihan 2005; Sangwine 1997). In the case of octonions, references to the analysis of 1- and 2-dimensional

signals appear in the literature (Grigoryan and Agaian 2018), but there is still no definition of discrete octonion Fourier transform of 3-dimensional signals.

The paper is organized as follows. In Sect. 2 we recall previous results concerning the octonion Fourier transform and introduce some important properties. In Sect. 3 we focus on the discrete-time LTI systems, which leads to the definition of the discrete-space octonion Fourier transform. Its well-posedness and some of its properties are the main part of Sect. 4. Sections 5 and 6 are devoted to the discrete octonion Fourier transform—its definition, properties and implementation aspects. The implementation of the DOFT algorithm and the results of the first tests are also presented there. The paper is concluded in Sect. 7 with a short discussion of obtained results.

2 Octonion Fourier transform and some properties

As mentioned earlier, the OFT of the real-valued function $u\colon \mathbb{R}^3 \to \mathbb{R}$ is given by the formula (1). In order for the integral (1) to exist, it is necessary for the function u to be at least integrable. However, in general, conditions of existence of the OFT are the same as for the classical (complex) Fourier transform (Błaszczyk 2020). Choice and order of imaginary units in the exponents is not accidental (see Błaszczyk and Snopek 2017). In our previous studies (see Błaszczyk and Snopek 2017), we proved the inverse formula (for continuous functions $u\colon \mathbb{R}^3 \to \mathbb{R}$ such that both u and its OFT are integrable):

$$u(\mathbf{x}) = \int_{\mathbb{R}^3} U(\mathbf{f}) \cdot e^{2\pi \mathbf{e}_4 f_3 x_3} \cdot e^{2\pi \mathbf{e}_2 f_2 x_2} \cdot e^{2\pi \mathbf{e}_1 f_1 x_1} \, \mathrm{d}\mathbf{f},$$

where $\mathbf{x} = (x_1, x_2, x_3)$, $\mathbf{f} = (f_1, f_2, f_3)$ and multiplication is performed from left to right. We have also proved some important features, among which there are those that will be useful in the analysis of discrete-variable signals (Błaszczyk 2020; Błaszczyk and Snopek 2017). Below, we quote their wording, assuming in each of these statements that the OFTs are well-defined and invertible. Let U be the OFT of the real-valued function u and let u_{x_i} denote the partial derivative of u with respect to x_i.

Theorem 1 (Shift Theorem) *Let U^α, U^β and U^γ denote the OFTs of $u(x_1 - \alpha, x_2, x_3)$, $u(x_1, x_2 - \beta, x_3)$ and $u(x_1, x_2, x_3 - \gamma)$, respectively. Then*

$$\begin{aligned}
U^\alpha(f_1, f_2, f_3) &= \cos(2\pi f_1 \alpha)\, U(f_1, f_2, f_3) - \sin(2\pi f_1 \alpha)\, U(f_1, -f_2, -f_3) \cdot \mathbf{e}_1,\\
U^\beta(f_1, f_2, f_3) &= \cos(2\pi f_2 \beta)\, U(f_1, f_2, f_3) - \sin(2\pi f_2 \beta)\, U(f_1, \ f_2, -f_3) \cdot \mathbf{e}_2,\\
U^\gamma(f_1, f_2, f_3) &= \cos(2\pi f_3 \gamma)\, U(f_1, f_2, f_3) - \sin(2\pi f_3 \gamma)\, U(f_1, \ f_2, \ f_3) \cdot \mathbf{e}_4.
\end{aligned}$$

Theorem 2 *Let $U^{\partial x_1}$, $U^{\partial x_2}$ and $U^{\partial x_3}$ denote the OFTs of u_{x_1}, u_{x_2} and u_{x_3}, respectively. Then*

$$\begin{aligned}
U^{\partial x_1}(f_1, f_2, f_3) &= U(f_1, -f_2, -f_3) \cdot (2\pi f_1 \mathbf{e}_1),\\
U^{\partial x_2}(f_1, f_2, f_3) &= U(f_1, \ f_2, -f_3) \cdot (2\pi f_2 \mathbf{e}_2),\\
U^{\partial x_3}(f_1, f_2, f_3) &= U(f_1, \ f_2, \ f_3) \cdot (2\pi f_3 \mathbf{e}_4).
\end{aligned}$$

It seems that in the above form these statements are of little use. It is worth noting, however, that on the one hand we have a theorem on Hermitian symmetry (Błaszczyk and Snopek 2017), and on the other hand we can reformulate the given formulas using the operation $\odot$. Then these theorems take the form known from classical theory.

Corollary 1 *Let U^α, U^β and U^γ denote the OFTs of $u(x_1-\alpha, x_2, x_3)$, $u(x_1, x_2-\beta, x_3)$ and $u(x_1, x_2, x_3-\gamma)$, respectively. Then*

$$U^\alpha(\mathbf{f}) = U(\mathbf{f}) \odot \mathrm{e}^{-\mathbf{E}_1 2\pi f_1 \alpha}, \quad U^\beta(\mathbf{f}) = U(\mathbf{f}) \odot \mathrm{e}^{-\mathbf{E}_2 2\pi f_2 \beta}, \quad U^\gamma(\mathbf{f}) = U(\mathbf{f}) \odot \mathrm{e}^{-\mathbf{E}_4 2\pi f_3 \gamma}.$$

Corollary 2 *Let $U^{\partial x_1}$, $U^{\partial x_2}$ and $U^{\partial x_3}$ denote the OFTs of u_{x_1}, u_{x_2} and u_{x_3}, respectively. Then*

$$U^{\partial x_1}(\mathbf{f}) = U(\mathbf{f}) \odot (2\pi f_1 \mathbf{E}_1), \quad U^{\partial x_2}(\mathbf{f}) = U(\mathbf{f}) \odot (2\pi f_2 \mathbf{E}_2), \quad U^{\partial x_3}(\mathbf{f}) = U(\mathbf{f}) \odot (2\pi f_3 \mathbf{E}_4).$$

Remark 1 The claims of the Corollaries 1 and 2 should be understood in a specific way. We will show it on the example of the last formula in Corollary 2. The OFT of the real-valued function u can be expressed as

$$U = U_0 + U_1\mathbf{e}_1 + U_2\mathbf{e}_2 + U_3\mathbf{e}_3 + U_4\mathbf{e}_4 + U_5\mathbf{e}_5 + U_6\mathbf{e}_6 + U_7\mathbf{e}_7,$$

where $U_0, \ldots, U_7 \colon \mathbb{R}^3 \to \mathbb{R}$ are real-valued functions (we omit the argument of the function so as not to lose readability of the formula). Then

$$U \cdot \mathbf{e}_4 = -U_4 - U_5\mathbf{e}_1 - U_6\mathbf{e}_2 - U_7\mathbf{e}_3 + U_1\mathbf{e}_4 + U_2\mathbf{e}_5 + U_3\mathbf{e}_6 + U_4\mathbf{e}_7.$$

On the other hand we can treat the octonion-valued function U as the 8-tuple of real-valued functions, which allows us to write this function as a $\mathbb{F}$-valued function:

$$U = U_0 + U_1\mathbf{E}_1 + U_2\mathbf{E}_2 + U_3\mathbf{E}_3 + U_4\mathbf{E}_4 + U_5\mathbf{E}_5 + U_6\mathbf{E}_6 + U_7\mathbf{E}_7.$$

Then

$$U \odot \mathbf{E}_4 = -U_4 - U_5\mathbf{E}_1 - U_6\mathbf{E}_2 - U_7\mathbf{E}_3 + U_1\mathbf{E}_4 + U_2\mathbf{E}_5 + U_3\mathbf{E}_6 + U_4\mathbf{E}_7,$$

which, again treating this expression as 8-tuple of real-valued functions, gives the statement of the Corollary. Other expressions should be understood analogously.

Proofs of these claims are based on direct calculations and we omit the details here. It should be remembered that the notation related to the operation $\odot$ serves only to increase the readability of performed operations and facilitate their interpretation. More importantly, it allows the direct use of the OFT for the analysis of LTI systems, which are described both by partial differential equations and difference equations (of three variables).

3 Discrete-time LTI systems

Consider linear time-invariant stationary system of three variables. We know from the classical signal and system theory that such systems are described by their impulse responses $h \colon \mathbb{R}^3 \to \mathbb{R}$ (also sometimes called Green functions) and then the input-output relation of signals $u \colon \mathbb{R}^3 \to \mathbb{R}$ and $v \colon \mathbb{R}^3 \to \mathbb{R}$, respectively, is given by the formula:

$$v(\mathbf{x}) = \int_{\mathbb{R}^3} u(\mathbf{y}) \cdot h(\mathbf{x}-\mathbf{y})\,\mathrm{d}\mathbf{y} = (u * h)(\mathbf{x}).$$

The function at the output of the system is therefore a convolution of the function on the input with the impulse response, which is schematically presented in the block form as in Fig. 3.

From the convolution-multiplication duality theorem for the classic Fourier transformation, it immediately follows that the following equality occurs:

$$\mathcal{F}_{\mathrm{CFT}}\{v\} = \mathcal{F}_{\mathrm{CFT}}\{u * h\} = \mathcal{F}_{\mathrm{CFT}}\{h\} \cdot \mathcal{F}_{\mathrm{CFT}}\{u\},$$

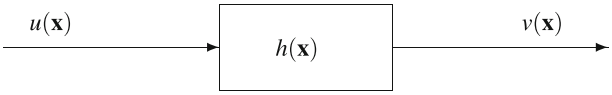

Fig. 3 3-D-LTI system (block representation)

and function $H = \mathcal{F}_{\mathrm{CFT}}\{h\}$ is called the frequency response of the system. In Błaszczyk (2019, 2020) we showed that the similar relation is valid also in case of octonion Fourier transform:

$$V(\mathbf{f}) = H_{\mathrm{OFT}}(\mathbf{f}) \odot U(\mathbf{f}),$$

where H_{OFT} is the octonion Fourier transform of the impuls response h (and therefore called the octonion frequency response) and we use the multiplication in the algebra of quadruple-complex numbers (in the sense described in Remark 1).

Thanks to these relations (and also Corollary 2), we can see that every linear partial differential equation with constant coefficients can be reduced to the algebraic equation (in the sense of multiplication in $\mathbb{F}$). In Błaszczyk (2019, 2020) we presented a few examples showing the possibilities of using this notation.

Analogous reasoning can be performed for discrete-time systems. They are described mostly by difference equations, i.e. equations of the form:

$$\sum_{i_1=0}^{M_1-1}\sum_{i_2=0}^{M_2-1}\sum_{i_3=0}^{M_3-1} a_{\mathbf{i}}\, u(\mathbf{n}-\mathbf{i}) = \sum_{j_1=0}^{N_1-1}\sum_{j_2=0}^{N_2-1}\sum_{j_3=0}^{N_3-1} b_{\mathbf{j}}\, v(\mathbf{n}-\mathbf{j}), \qquad a_{(0,0,0)} \neq 0$$

where $u, v\colon \mathbb{N}^3 \to \mathbb{R}$ are unknown sequences indexed by a 3-D discrete variable, $a_{\mathbf{i}}, b_{\mathbf{j}} \in \mathbb{R}$, $\mathbf{n} = (n_1, n_2, n_3)$, $\mathbf{i} = (i_1, i_2, i_3)$ and $\mathbf{j} = (j_1, j_2, j_3)$. Using the octonion Fourier transform and Theorem 1, we can write this equality in a different form. However, only by composing Corollary 1 (and treating octonions as 8-tuples of real numbers, and therefore as elements of $\mathbb{F}$) we can reduce this relation to that which we know from classical theory:

$$\begin{aligned} U(\mathbf{f}) \odot \left(\sum_{i_1=0}^{M_1-1}\sum_{i_2=0}^{M_2-1}\sum_{i_3=0}^{M_3-1} a_{\mathbf{i}}\, \mathrm{e}^{-\mathbf{E}_1 2\pi f_1 i_1} \odot \mathrm{e}^{-\mathbf{E}_2 2\pi f_2 i_2} \odot \mathrm{e}^{-\mathbf{E}_4 2\pi f_3 i_3} \right) \\ = V(\mathbf{f}) \odot \left(\sum_{j_1=0}^{N_1-1}\sum_{j_2=0}^{N_2-1}\sum_{j_3=0}^{N_3-1} b_{\mathbf{j}}\, \mathrm{e}^{-\mathbf{E}_1 2\pi f_1 j_1} \odot \mathrm{e}^{-\mathbf{E}_2 2\pi f_2 j_2} \odot \mathrm{e}^{-\mathbf{E}_4 2\pi f_3 j_3} \right). \end{aligned}$$

It can be shown with direct calculations that

$$\mathrm{e}^{-\mathbf{E}_1 2\pi f_1 i_1} \odot \mathrm{e}^{-\mathbf{E}_2 2\pi f_2 i_2} \odot \mathrm{e}^{-\mathbf{E}_4 2\pi f_3 i_3} = \mathrm{e}^{-\mathbf{e}_1 2\pi f_1 i_1} \cdot \mathrm{e}^{-\mathbf{e}_2 2\pi f_2 i_2} \cdot \mathrm{e}^{-\mathbf{e}_4 2\pi f_3 i_3},$$

where the multiplication in the octonion algebra is performed from left to right. The above equality should be understood in the sense described in Remark 1, i.e. as the equality of 8-tuples of real numbers. Therefore we have

$$\begin{aligned} U(\mathbf{f}) \odot \left(\sum_{i_1=0}^{M_1-1}\sum_{i_2=0}^{M_2-1}\sum_{i_3=0}^{M_3-1} a_{\mathbf{i}}\, \mathrm{e}^{-\mathbf{E}_1 2\pi f_1 i_1} \odot \mathrm{e}^{-\mathbf{E}_2 2\pi f_2 i_2} \odot \mathrm{e}^{-\mathbf{E}_4 2\pi f_3 i_3} \right) \\ = V(\mathbf{f}) \odot \left(\sum_{j_1=0}^{N_1-1}\sum_{j_2=0}^{N_2-1}\sum_{j_3=0}^{N_3-1} b_{\mathbf{j}}\, \mathrm{e}^{-\mathbf{E}_1 2\pi f_1 j_1} \odot \mathrm{e}^{-\mathbf{E}_2 2\pi f_2 j_2} \odot \mathrm{e}^{-\mathbf{E}_4 2\pi f_3 j_3} \right), \end{aligned}$$

where expressions in brackets are some octonion counterparts to discrete Fourier transforms of vectors $\mathbf{a} = (a_{\mathbf{i}})$, $\mathbf{b} = (b_{\mathbf{j}})$ indexed by 3-D discrete variables.

4 Discrete-space octonion Fourier transform

The above considerations lead to the definition of the octonion counterpart of the discrete Fourier transform of real-valued sequences.

Definition 1 Let $\mathbf{a}\colon \mathbb{N}^3 \to \mathbb{R}$ be a sequence indexed by a 3-D discrete variable and let $\mathbf{a} = (a_{\mathbf{i}})$, $\mathbf{i} = (i_1, i_2, i_3)$. *Octonion Fourier transform of the sequence* $\mathbf{a}$ is defined by the formula

$$A_{\mathrm{OFT}}(\mathbf{f}) = \sum_{\mathbf{i}\in\mathbb{N}^3} a_{\mathbf{i}} \cdot \mathrm{e}^{-\mathbf{e}_1 2\pi f_1 i_1} \cdot \mathrm{e}^{-\mathbf{e}_2 2\pi f_2 i_2} \cdot \mathrm{e}^{-\mathbf{e}_4 2\pi f_3 i_3}, \quad \mathbf{f} = (f_1, f_2, f_3) \in \left(-\frac{1}{2}, \frac{1}{2}\right)^3,$$

where multiplication is performed from left to right.

The above definition is a three-dimensional equivalent of the Fourier transformation of discrete time (*discrete-time Fourier transform*—DTFT), in relation to which the given formula can be abbreviatied as DSOFT (*discrete-space octonion Fourier transform*). Like its classic equivalent, DSOFT is a periodic function with relation to each variable, with a period of 1. Using the methods presented in the proof of the inverse theorem (Błaszczyk and Snopek 2017), the following formula can be derived for the inverse transform.

Theorem 3 *Let* $\mathbf{a}\colon \mathbb{N}^3 \to \mathbb{R}$ *be a sequence indexed by a 3-D discrete variable and let* $\mathbf{a} = (a_{\mathbf{i}})$, $\mathbf{i} = (i_1, i_2, i_3)$, *be square-summable. Then*

$$a_{\mathbf{i}} = \int_{\left(-\frac{1}{2}, \frac{1}{2}\right)^3} A_{\mathrm{OFT}}(\mathbf{f}) \cdot \mathrm{e}^{\mathbf{e}_4 2\pi f_3 i_3} \cdot \mathrm{e}^{\mathbf{e}_2 2\pi f_2 i_2} \cdot \mathrm{e}^{\mathbf{e}_1 2\pi f_1 i_1} \, \mathrm{d}\mathbf{f},$$

where multiplication is performed from left to right.

This theorem can be generalized to octonion-valued sequences, proving it with the same methods as in Błaszczyk (2018, 2020). From simple calculations the proof of the following theorem on the transformation of the rescaled function follows.

Theorem 4 *Let* $\mathbf{a} = (a_{\mathbf{i}})\colon \mathbb{N}^3 \to \mathbb{R}$ *and let* A *denote the DSOFT of* $\mathbf{a}$. *Let* $k_1, k_2, k_3 \in \mathbb{N}_+$, *define* $\mathbf{b}\colon \mathbb{N}^3 \to \mathbb{R}$ *by the formula:*

$$b_{(i_1,i_2,i_3)} = \begin{cases} a_{(i_1/k_1, i_2/k_2, i_3/k_3)} & \text{if } k_1|i_1,\ k_2|i_2 \text{ and } k_3|i_3, \\ 0 & \text{otherwise} \end{cases}$$

and let B *denote the DSOFT of* $\mathbf{b}$. *Then*

$$B(f_1, f_2, f_3) = A(k_1 f_1, k_2 f_2, k_3 f_3).$$

Proof From straightforward calculations we have that

$$\begin{aligned} B(f_1, f_2, f_3) &= \sum_{\mathbf{i}\in\mathbb{N}^3} b_{\mathbf{i}} \cdot \mathrm{e}^{-\mathbf{e}_1 2\pi f_1 i_1} \cdot \mathrm{e}^{-\mathbf{e}_2 2\pi f_2 i_2} \cdot \mathrm{e}^{-\mathbf{e}_4 2\pi f_3 i_3} \\ &= \sum_{\mathbf{j}\in\mathbb{N}^3} a_{\mathbf{j}} \cdot \mathrm{e}^{-\mathbf{e}_1 2\pi f_1 k_1 j_1} \cdot \mathrm{e}^{-\mathbf{e}_2 2\pi f_2 k_2 j_2} \cdot \mathrm{e}^{-\mathbf{e}_4 2\pi f_3 k_3 j_3} = A(k_1 f_1, k_2 f_2, k_3 f_3), \end{aligned}$$

which concludes the proof. □

It is worth noting that proofs of other properties of the octonion Fourier transformation (of functions of the continuous variable) proceeded in the same way as proofs of equivalents of these properties for the classic Fourier transform. Differences in statements of those theorems resulted only from the properties of multiplication of octonions, and, as a consequence, of operations on exponential functions in the kernel of transformation. Therefore, the equivalents of these properties for DSOFT will look the same, and their proofs will be very similar. We will therefore omit the details and quote only the statements.

Theorem 5 *Let* $\mathbf{a} = (a_{\mathbf{i}}) \colon \mathbb{N}^3 \to \mathbb{R}$ *and let A denote the DSOFT of* $\mathbf{a}$. *Moreover, let* $f_0 \in \mathbb{R}$ *and denote* $\mathbf{a}^{\cos,k} = (a_{\mathbf{i}})$, $a_{\mathbf{i}}^{\cos,k} = a_{\mathbf{i}} \cdot \cos(2\pi f_0 i_k)$, *and* $A^{\cos,k}$ *as the DSOFT of* $\mathbf{a}^{\cos,k}$, $k = 1, 2, 3$. *Then*

$$A^{\cos,1}(f_1, f_2, f_3) = \Big(A(f_1 + f_0, f_2, f_3) + A(f_1 - f_0, f_2, f_3)\Big) \cdot \frac{1}{2},$$
$$A^{\cos,2}(f_1, f_2, f_3) = \Big(A(f_1, f_2 + f_0, f_3) + A(f_1, f_2 - f_0, f_3)\Big) \cdot \frac{1}{2},$$
$$A^{\cos,3}(f_1, f_2, f_3) = \Big(A(f_1, f_2, f_3 + f_0) + A(f_1, f_2, f_3 - f_0)\Big) \cdot \frac{1}{2}.$$

Theorem 6 *Let* $\mathbf{a} = (a_{\mathbf{i}}) \colon \mathbb{N}^3 \to \mathbb{R}$ *and let A denote the DSOFT of* $\mathbf{a}$. *Moreover, let* $f_0 \in \mathbb{R}$ *and denote* $\mathbf{a}^{\sin,k} = (a_{\mathbf{i}})$, $a_{\mathbf{i}}^{\sin,k} = a_{\mathbf{i}} \cdot \sin(2\pi f_0 i_k)$, *and* $A^{\cos,k}$ *as the DSOFT of* $\mathbf{a}^{\sin,k}$, $k = 1, 2, 3$. *Then*

$$A^{\sin,1}(f_1, f_2, f_3) = \Big(A(f_1 + f_0, -f_2, -f_3) - A(f_1 - f_0, -f_2, -f_3)\Big) \cdot \frac{\mathbf{e}_1}{2},$$
$$A^{\sin,2}(f_1, f_2, f_3) = \Big(A(f_1, f_2 + f_0, -f_3) - A(f_1, f_2 - f_0, -f_3)\Big) \cdot \frac{\mathbf{e}_2}{2},$$
$$A^{\sin,3}(f_1, f_2, f_3) = \Big(A(f_1, f_2, f_3 + f_0) - A(f_1, f_2, f_3 - f_0)\Big) \cdot \frac{\mathbf{e}_4}{2}.$$

From the reasoning in the previous section, the shift theorem also immediately follows.

Theorem 7 *Let* $\mathbf{a} = (a_{\mathbf{i}}) \colon \mathbb{N}^3 \to \mathbb{R}$ *and let A denote the DSOFT of* $\mathbf{a}$. *Moreover, let* $\alpha, \beta, \gamma \in \mathbb{Z}$ *and denote* $\mathbf{a}^{\alpha} = (a_{\mathbf{i}}^{\alpha})$, $a^{\alpha}_{(i_1,i_2,i_3)} = a_{(i_1-\alpha,i_2,i_3)}$, $\mathbf{a}^{\beta} = (a_{\mathbf{i}}^{\beta})$, $a^{\beta}_{(i_1,i_2,i_3)} = a_{(i_1,i_2-\beta,i_3)}$, $\mathbf{a}^{\gamma} = (a_{\mathbf{i}}^{\gamma})$, $a^{\gamma}_{(i_1,i_2,i_3)} = a_{(i_1,i_2,i_3-\gamma)}$. *Let* A^{ℓ} *denote the DSOFT of* $\mathbf{a}^{\ell}$, *where* $\ell = \alpha, \beta, \gamma$. *Then*

$$A^{\alpha}(f_1, f_2, f_3) = \cos(2\pi f_1 \alpha) A(f_1, f_2, f_3) - \sin(2\pi f_1 \alpha) A(f_1, -f_2, -f_3) \cdot \mathbf{e}_1$$
$$A^{\beta}(f_1, f_2, f_3) = \cos(2\pi f_2 \beta) A(f_1, f_2, f_3) - \sin(2\pi f_2 \beta) A(f_1, f_2, -f_3) \cdot \mathbf{e}_2$$
$$A^{\gamma}(f_1, f_2, f_3) = \cos(2\pi f_3 \gamma) A(f_1, f_2, f_3) - \sin(2\pi f_3 \gamma) A(f_1, f_2, f_3) \cdot \mathbf{e}_4.$$

As we mentioned earlier, deriving the DSOFT definition, it can be used to analyze discrete systems described by difference equations. It is also easy to see that these methods will be used in the analysis of finite difference schemes for partial differential equations. The theory considered so far mainly used classic Fourier transforms (discrete), which were applied to a variable interpreted as space (one- or two-dimensional), but by defining an octonion transformation we can try to transform the whole scheme, both for time and space.

5 Discrete octonion Fourier transform

In the previous section we considered signals (sequences) of infinite length. However, in practice, finite-length signals are usually found, which, as in the classical case, leads to the definition of discrete octonion Fourier transform. Similarly to the complex and quaternion case, the following definition has its basis in the discretization of the frequency domain in discrete-space octonion Fourier transform.

Definition 2 Let $\mathbf{a}\colon [N_1] \times [N_2] \times [N_3] \to \mathbb{R}$, $[N_i] = \{0, \ldots, N_i - 1\}$, be a finite-length sequence indexed by a 3-D discrete variable and let $\mathbf{a} = (a_{\mathbf{n}})$, $\mathbf{n} = (n_1, n_2, n_3)$. *Discrete octonion Fourier transform* (DOFT) $\mathbf{A}_{\mathrm{OFT}} = (A_{\mathrm{OFT},\mathbf{k}})$ of $\mathbf{a}$ is defined by the formula

$$A_{\mathrm{OFT},\mathbf{k}} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \sum_{n_3=0}^{N_3-1} a_{(n_1,n_2,n_3)} \cdot \mathrm{e}^{-\mathbf{e}_1 2\pi k_1 n_1/N_1} \cdot \mathrm{e}^{-\mathbf{e}_2 2\pi k_2 n_2/N_2} \cdot \mathrm{e}^{-\mathbf{e}_4 2\pi k_3 n_3/N_3}, \quad (2)$$

where $\mathbf{k} = (k_1, k_2, k_3) \in [N_1] \times [N_2] \times [N_3]$ and multiplication is performed from left to right.

This is the direct octonion equivalent of the 3-D discrete Fourier transformation. The formula for inverse transformation is proved analogously to the corresponding formula for inverse OFT (Błaszczyk 2020). In particular, the following equality is used:

$$\frac{1}{N} \sum_{k=0}^{N-1} \mathrm{e}^{-\mathbf{e}_i 2\pi kn/N} \cdot \mathrm{e}^{\mathbf{e}_i 2\pi km/N} = \delta_{m,n} = \begin{cases} 1 & \text{if } m = n, \\ 0 & \text{if } m \neq n, \end{cases}$$

valid for every octonion imaginary unit $\mathbf{e}_i$, $i = 1, \ldots, 7$. We omit the proof and only quote the statement of the inverse-transform theorem.

Theorem 8 *Let* $\mathbf{a}\colon [N_1] \times [N_2] \times [N_3] \to \mathbb{R}$, $[N_i] = \{0, \ldots, N_i - 1\}$, *be a finite-length sequence indexed by a 3-D discrete variable and let* $\mathbf{a} = (a_{\mathbf{n}})$, $\mathbf{n} = (n_1, n_2, n_3)$. *If* $\mathbf{A}_{\mathrm{OFT}} = (A_{\mathrm{OFT},\mathbf{k}})$ *is the DOFT of* $\mathbf{a}$, *then*

$$a_{\mathbf{n}} = \frac{1}{N_1 N_2 N_3} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \sum_{k_3=0}^{N_3-1} A_{\mathrm{OFT},(k_1,k_2,k_3)} \cdot \mathrm{e}^{\mathbf{e}_4 2\pi k_3 n_3/N_3} \cdot \mathrm{e}^{\mathbf{e}_2 2\pi k_2 n_2/N_2} \cdot \mathrm{e}^{\mathbf{e}_1 2\pi k_1 n_1/N_1},$$

where multiplication is performed from left to right.

From a computational point of view, consideration should be given to the possibility of implementing a fast DOFT calculation version. In the quaternion case, various solutions to this problem have been proposed—direct implementation of the quaternion version of the Fast Fourier Transform algorithm (FFT) and the use of the complex (original) version of this algorithm. The latter option turned out to be better and less computationally demanding (Ell et a. 2014).

In the case of octonions, one can do the same. In Błaszczyk (2020) we proved that the octonion Fourier transformation of octonion-valued functions can be calculated using the classical Fourier transformation. This was formulated in the following statement (for proof see Błaszczyk 2020).

Theorem 9 *Let* $u\colon \mathbb{R}^3 \to \mathbb{O}$ *and* $u = v_0+v_1\mathbf{e}_2+v_2\mathbf{e}_4+v_3\mathbf{e}_2\cdot\mathbf{e}_4$, *where* $v_0, \ldots, v_3\colon \mathbb{R}^3 \to \mathbb{C}$. *If* U_{OFT} *is the OFT of* u, *then*

$$\begin{aligned} U_{\mathrm{OFT}}(f_1, f_2, f_3) = V_0(\, f_1,\ f_2,\ f_3) &+ V_1(-f_1,\ f_2, -f_3)\cdot\mathbf{e}_2 \\ &+ V_2(-f_1, -f_2,\ f_3)\cdot\mathbf{e}_4 + V_3(\, f_1, -f_2, -f_3)\cdot\mathbf{e}_2\cdot\mathbf{e}_4, \end{aligned}$$

where V_i *is the OFT of* v_i, $i = 0, \ldots, 3$, *and*

$$\begin{aligned} V_i(f_1, f_2, f_3) = &\frac{1}{4}\big(\widehat{v_i}(f_1, f_2, f_3)\cdot(1-\mathbf{e}_3) + \widehat{v_i}(f_1, -f_2, f_3)\cdot(1+\mathbf{e}_3)\big)\cdot(1-\mathbf{e}_5) \\ &+ \frac{1}{4}\big(\widehat{v_i}(f_1, f_2, -f_3)\cdot(1-\mathbf{e}_3) + \widehat{v_i}(f_1, -f_2, -f_3)\cdot(1+\mathbf{e}_3)\big)\cdot(1+\mathbf{e}_5), \end{aligned}$$

where $\widehat{v_i}$ *is the classical (complex) Fourier transform of* v_i, $i = 0, \ldots, 3$.

It is important to notice that the field of complex numbers $\mathbb{C}$ used in the abovementioned theorem is the specific subfield of the octonion algebra, i.e. $\mathbb{C} = \{x_0+x_1\mathbf{e}_1 \in \mathbb{O}\colon x_0, x_1 \in \mathbb{R}\}$. Consequently, only the imaginary units $\mathbf{e}_2$ and $\mathbf{e}_4$ appear in the thesis. Equivalent versions of the above theorem can be derived which take into account other pairs of imaginary units.

Analogous theorems for calculating the inverse transformation are also known (Błaszczyk 2020, Theorem 5 and Corollary 2). Due to the long formulation of the theorem, we omit it here. These claims remain true also for discrete transformations. The operation of changing the sign of a variable should be understood in the sense of modulo operations, as in the classic case—DOFT, just like DFT, can be treated as a periodic function.

Thanks to this, it is possible to use all the advantages of the FFT algorithm, with a small additional calculation effort—the octonion FFT algorithm for functions with octonion values requires the calculation of four transforms of different functions with complex values.

Tools for operations on hypercomplex numbers are available in many programming environments, including in MATLAB®. MATLAB® environment is one of the more popular tools supporting the work of engineers and packages expanding programming capabilities in this environment appear quite often. The `qtfm` package developed by the team of S. Sangwine and N. Le Bihan Sangwine and Bihan (2005) focuses on numerical calculations in quaternion algebra (not only basic arithmetic operations are available, but also highly developed tools for calculating quaternion Fourier transforms), but on the other hand, it also allows simple operations in octonion algebra. However, it lacks more advanced features that would give users the opportunity to calculate octonion Fourier transforms.

The `qtfm` package has been extended by, among others, five additional functions

- `doft3` and `idoft3` (forward and inverse DOFT using the direct formula),
- `offt3` and `iofft3` (forward and inverse DOFT using the FFT algorithm),
- `fftreflection` (symmetrical reflection of functions relative to the indicated axes).

The functions have been implemented using the `qtfm` package syntax. Octonions are entered in it, among others as

```
octonion(r0, r1, r2, r3, r4, r5, r6, r7),
```

where `r0`, …, `r7` are numbers or matrixes of numbers. It is also possible to automatically project real numbers, complex numbers and quaternions to the appropriate octonions, using the fact that they are created in the Cayley-Dickson construction (this is done using the available function `cd`).

The implementation of the forward and inverse discrete octonion Fourier transformation was performed in two ways—using the direct formula (2) and using Theorem 9, where first

the classical (complex) Fourier transforms are calculated, and then they are combined in an appropriate manner. In the case of the direct algorithm, in order to avoid problems with multiplication of octonions, the matrix representation of multiplication presented in Tian (2000) was used. One of the important elements of the octonion FFT algorithm is the ability to symmetrically reflect functions with relation to the respective axes. This has been implemented in the `fftreflection(X, A)` function, where `X` is a mirrored function (matrix), and `A` is a vector that indicates which variables to reflect (e.g. `[1,3]` means that `X` is mirrored with respect to 1st and 3rd variable). It uses the convention that MATLAB® adopts with the FFT algorithm – zero frequencies always appear on the first coordinates of the matrix (in each dimension), and the discrete Fourier transform is a periodic function.

Implemented functions can be found in the GitHub repository:

https://github.com/blaszczykl/matlab-octonions and the code is also included in Appendix 1 (Listings 1–5). This is the original code developed by the author, however, it requires the `qtfm` package to work correctly (it is based on the syntax used in this package).

The correctness of implemented functions can be tested in two ways. First, it was checked whether applying the reverse transformation to the result of the forward transformation returned the original matrix. This check was done for both versions of the algorithm by generating random $N \times N \times N$ octonion matrices, where $N = 4$ and each coordinate was generated from a uniform distribution over the interval [0, 1] (for details see Listing 7). In both cases, the maximum relative error did not exceed $2.2 \cdot 10^{-11}\%$, which can be treated as a numerical approximation error.

It is worth noting the (expected) problem of calculation time for various quantities of N. In the case of the direct algorithm, the execution time is proportional to N^6, while for the FFT algorithm the time is proportional to $N^3 \log N$. This is not a surprising feature, it also applies to the classical DFT, however, due to operations on octonions, the time needed to perform calculations may be larger than in the case of the classical one. It can be seen that this issue is important.

To check whether an error was made when implementing the algorithm using Theorem 9 and whether it actually returns a discrete OFT, it was assumed that the direct algorithm returns the correct result and then it was compared with the result returned by the octonion FFT. In this case, the error did not exceed $1.5 \cdot 10^{-12}\%$, which suggests that the implementation is correct.

6 Symmetry properties of the DOFT

As in the case of continuous OFT, most discrete classical equivalents of Fourier transform properties can be proved for discrete OFT. One of the first results for continuous OFT was to show the equivalent of Hermitian symmetry (Błaszczyk and Snopek 2017, Theorem 4.6). In the discrete case, the following theorem can be proved by repeating the reasoning presented in Błaszczyk and Snopek (2017).

Theorem 10 *Let* $\mathbf{a}\colon [N_1] \times [N_2] \times [N_3] \to \mathbb{R}$, $[N_i] = \{0, \ldots, N_i - 1\}$, *be a finite-length sequence indexed by a 3-D discrete variable and let* $\mathbf{a} = (a_{\mathbf{n}})$, $\mathbf{n} = (n_1, n_2, n_3)$. *If* $\mathbf{A}_{\mathrm{OFT}} = (A_{\mathrm{OFT},\mathbf{n}})$ *is the DOFT of* $\mathbf{a}$, *then*

$$A_{\mathrm{OFT},(m_1,n_2,n_3)} = -\alpha_{7,5,3,1}(A_{\mathrm{OFT},(n_1,n_2,n_3)}),$$
$$A_{\mathrm{OFT},(n_1,m_2,n_3)} = -\alpha_{7,6,3,2}(A_{\mathrm{OFT},(n_1,n_2,n_3)}),$$
$$A_{\mathrm{OFT},(n_1,n_2,m_3)} = -\alpha_{7,6,5,4}(A_{\mathrm{OFT},(n_1,n_2,n_3)}),$$

where $m_i = (-n_i \mod N_i)$, $\alpha_{i_1,\ldots,i_4} = \alpha_{i_1} \circ \ldots \circ \alpha_{i_4}$ *and* $\alpha_i(o) = -\mathbf{e}_i \cdot o \cdot \mathbf{e}_i$.

It should be noted that the function $-\alpha_{i_1,\ldots,i_4}$, introduced in Błaszczyk and Snopek (2017), changes the sign of four imaginary units of an octonion, i.e. $\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_4}$. The above theorem should be interpreted so that the individual imaginary parts are even or odd with relation to the respective variables (in the sense of modulo operations).

The function $\alpha_{i_1,\ldots,i_4}$ has also been implemented on the basis of the `qtfm` package as a function `alpha(o, i1, i2, i3, i4)` (see Listing 6), where `o` is the transformed octonion (or matrix of octonions) and `i1`, …, `i4` are four (different) indices of imaginary parts of `o` whose sign should be changed. The tests that could be performed thanks to this (presented in Listing 7) illustrate Theorem 10 and show that indeed DOFT of real-value matrices has the mentioned symmetry properties.

7 Discussion and conclusions

The results presented show that discrete Fourier transforms can be generalized to the case of higher order algebras (e.g. octonions). What's more, using the properties of algebra of quadruple-complex numbers, this generalization can lead to a very similar form.

One could ask the question why it is worth working in two different algebras in parallel instead of just working in the algebra of quadruple-complex numbers $\mathbb{F}$. While going through calculations in $\mathbb{F}$ facilitates the interpretation of results, there are more and more papers in the literature devoted to the use of octonion algebra (Gao and Lam 2014; Grigoryan and Agaian 2018; Hahn and Snopek 2016; Klco et al. 2017; Lazendić et al. 2018a, b; Lian 2019; Popa 2016, 2018; Sheng et al. 2018; Snopek 2015; Wang et al. 2017). Therefore, it seems important to develop tools enabling work in this algebra as well.

The properties of discrete octonion Fourier transforms show that they can be used without difficulty for the analysis of difference equations, as well as for the analysis of finite difference schemes for differential equations. Detailed work in this area remains in the plans for further actions, as well as the further development of this theory.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

MATLAB® files

Listing 1 offt3.m

```
function Y = offt3(X)
% OFFT3 Discrete octonion Fourier transform calculated with FFT algorithm
```

```
%
% This function calculates the discrete octonion Fourier transform of
% 3-dimensional octonion-valued matrix X using classical FFT algorithm. It
% uses the convention that zero frequencies appear on the first coordinates
% of the matrix. In order to shift the zero frequency to the middle of the
% matrix, the fftshift function should be used.
%
% See also: IOFFT3, FFTSHIFT

narginchk(1, 1), nargoutchk(0, 1)

if ~isreal(X)
    error(['The transformed matrix must have components that are ', ...
        'real octonions.']);
end

if ndims(X) ~= 3
    error('The transformed matrix must be 3-dimensional.');
end

e2 = octonion(0,0,1,0,0,0,0,0);
e4 = octonion(0,0,0,0,1,0,0,0);

Y = octonion(zeros(size(X)));
for i = 1:4
    Ca = fftn(complex(part(X,2*i-1), part(X,2*i)));
        Cb = fftreflection(Ca, 2);
        Cc = fftreflection(Ca, 3);
        Cd = fftreflection(Cb, 3);
    V = octonion( real(Ca) + real(Cb) + real(Cc) + real(Cd),...
                  imag(Ca) + imag(Cb) + imag(Cc) + imag(Cd),...
                  imag(Ca) - imag(Cb) + imag(Cc) - imag(Cd),...
                 -real(Ca) + real(Cb) - real(Cc) + real(Cd),...
                  imag(Ca) + imag(Cb) - imag(Cc) - imag(Cd),...
                 -real(Ca) - real(Cb) + real(Cc) + real(Cd),...
                 -real(Ca) + real(Cb) + real(Cc) - real(Cd),...
                 -imag(Ca) + imag(Cb) + imag(Cc) - imag(Cd));
    switch i
        case 1
            Y = Y +  V;
        case 2
            Y = Y +  fftreflection(V,[1,3]) * e2;
        case 3
            Y = Y +  fftreflection(V,[1,2]) * e4;
        case 4
            Y = Y + (fftreflection(V,[2,3]) * e2) * e4;
    end
end
Y = Y / 4;
```

Listing 2 iofft3.m

```
function Y = iofft3(X)
% IOFFT3 Inverse discrete octonion Fourier transform calculated with FFT
% algorithm
%
% This function calculates the inverse discrete octonion Fourier transform
% of 3-dimensional octonion-valued matrix X using classical FFT algorithm.
% It uses the convention that zero frequencies appear on the first
% coordinates of the matrix. In order to shift the zero frequency back to
% the first coordinates, the ifftshift function should be applied.
%
% See also: OFFT3, IFFTSHIFT

narginchk(1, 1), nargoutchk(0, 1)
```

```
if ~isreal(X)
    error(['The transformed matrix must have components that are ', ...
        'real octonions.']);
end

if ndims(X) ~= 3
    error('The transformed matrix must be 3-dimensional.');
end

e1 = octonion(0,1,0,0,0,0,0,0);
e2 = octonion(0,0,1,0,0,0,0,0);

Y = octonion(zeros(size(X)));
for i = 1:4
    if i == 1 || i == 4
        Ca = ifftn(complex( part(X,i), part(X,i+4)));
    else
        Ca = ifftn(complex( part(X,i),-part(X,i+4)));
    end
        Cb = fftreflection(Ca, 2);
        Cc = fftreflection(Ca, 1);
        Cd = fftreflection(Cb, 1);
    V = octonion( real(Ca) + real(Cb) + real(Cc) + real(Cd),...
                  imag(Ca) + imag(Cb) - imag(Cc) - imag(Cd),...
                  imag(Ca) - imag(Cb) + imag(Cc) - imag(Cd),...
                  real(Ca) - real(Cb) - real(Cc) + real(Cd),...
                  imag(Ca) + imag(Cb) + imag(Cc) + imag(Cd),...
                  real(Ca) + real(Cb) - real(Cc) - real(Cd),...
                  real(Ca) - real(Cb) + real(Cc) - real(Cd),...
                  imag(Ca) - imag(Cb) - imag(Cc) + imag(Cd));
    switch i
        case 1
            Y = Y +  V;
        case 2
            Y = Y +  fftreflection(V,[2,3]) * e1;
        case 3
            Y = Y +  fftreflection(V,[1,3]) * e2;
        case 4
            Y = Y + (fftreflection(V,[1,2]) * e1) * e2;
    end
end
Y = Y / 4;
```

Listing 3 fftreflection.m

```
function Y = fftreflection(X, A)
% FFTREFLECTION Reflect the OFT spectrum with respect to some variable
%
% This function rearranges an octonion Fourier transform X by reflecting
% the spectrum with respect to variables with indices in vector A. It
% uses the convention that zero frequencies appear on the first coordinates
% of the matrix.
%
% See also: OFFT3, IOFFT3

narginchk(2, 2), nargoutchk(0, 1)

if ndims(X) ~= 3
    error('The transformed matrix must be 3-dimensional.');
end

if ~all(mod(A,1) == 0) || ~all(A >= 1) || ~all(A <= 3)
    error('A must be an array containing only 1s, 2s and 3s.');
end

Y = X;
```

```
for dim = A
    switch dim
        case 1
            Y(2:end,:,:) = flip(Y(2:end,:,:), 1);
        case 2
            Y(:,2:end,:) = flip(Y(:,2:end,:), 2);
        case 3
            Y(:,:,2:end) = flip(Y(:,:,2:end), 3);
    end
end
```

Listing 4 doft3.m

```
function Y = doft3(X)
% DOFT3 Discrete octonion Fourier transform calculated with direct formula
%
% This function calculates the discrete octonion Fourier transform of
% 3-dimensional octonion-valued matrix X using direct formula. It uses the
% convention that zero frequencies appear on the first coordinates of the
% matrix. In order to shift the zero frequency to the middle of the matrix,
% the fftshift function should be used.
%
% See also: IDOFT3, FFTSHIFT

narginchk(1, 1), nargoutchk(0, 1)

if ~isreal(X)
    error(['The transformed matrix must have components that are ', ...
        'real octonions.']);
end

if ndims(X) ~= 3
    error('The transformed matrix must be 3-dimensional.');
end

[N1, N2, N3] = size(X);

Y = octonion(zeros(N1,N2,N3));

for k1 = 0:N1-1, for k2 = 0:N2-1, for k3 = 0:N3-1
    y = zeros(8,1);
    for n1 = 0:N1-1, for n2 = 0:N2-1, for n3 = 0:N3-1
        for i = 1:8, x(i,1) = part(X(n1+1,n2+1,n3+1),i); end
        e1c = cos(2*pi*k1*n1/N1); e1s = -sin(2*pi*k1*n1/N1);
        e2c = cos(2*pi*k2*n2/N2); e2s = -sin(2*pi*k2*n2/N2);
        e4c = cos(2*pi*k3*n3/N3); e4s = -sin(2*pi*k3*n3/N3);
        E1 = diag(e1c*ones(8,1));
        E1(1,2) = -e1s; E1(2,1) =  e1s; E1(3,4) =  e1s; E1(4,3) = -e1s;
        E1(5,6) =  e1s; E1(6,5) = -e1s; E1(7,8) = -e1s; E1(8,7) =  e1s;
        E2 = diag(e2c*ones(8,1));
        E2(1,3) = -e2s; E2(2,4) = -e2s; E2(3,1) =  e2s; E2(4,2) =  e2s;
        E2(5,7) =  e2s; E2(6,8) =  e2s; E2(7,5) = -e2s; E2(8,6) = -e2s;
        E4 = diag(e4c*ones(8,1));
        E4(1,5) = -e4s; E4(2,6) = -e4s; E4(3,7) = -e4s; E4(4,8) = -e4s;
        E4(5,1) =  e4s; E4(6,2) =  e4s; E4(7,3) =  e4s; E4(8,4) =  e4s;
        y = y + E4 * (E2 * (E1 * x));
    end; end; end
    Y(k1+1,k2+1,k3+1) = octonion(y(1),y(2),y(3),y(4),y(5),y(6),y(7),y(8));
end; end; end
```

Listing 5 idoft3.m

```
function Y = idoft3(X)
% IDOFT3 Inverse discrete octonion Fourier transform calculated with direct
% formula
%
```

```
% This function calculates the inverse discrete octonion Fourier transform
% of 3-dimensional octonion-valued matrix X using direct formula. It uses
% the convention that zero frequencies appear on the first coordinates of
% the matrix. In order to shift the zero frequency back to the first
% coordinates, the ifftshift function should be applied.
%
% See also: DOFT3, IFFTSHIFT

narginchk(1, 1), nargoutchk(0, 1)

if ~isreal(X)
    error(['The transformed matrix must have components that are ', ...
        'real octonions.']);
end

if ndims(X) ~= 3
    error('The transformed matrix must be 3-dimensional.');
end

[N1, N2, N3] = size(X);

Y = octonion(zeros(N1,N2,N3));

for k1 = 0:N1-1, for k2 = 0:N2-1, for k3 = 0:N3-1
    y = zeros(8,1);
    for n1 = 0:N1-1, for n2 = 0:N2-1, for n3 = 0:N3-1
        for i = 1:8, x(i,1) = part(X(n1+1,n2+1,n3+1),i); end
        e1c = cos(2*pi*k1*n1/N1); e1s = sin(2*pi*k1*n1/N1);
        e2c = cos(2*pi*k2*n2/N2); e2s = sin(2*pi*k2*n2/N2);
        e4c = cos(2*pi*k3*n3/N3); e4s = sin(2*pi*k3*n3/N3);
        E1 = diag(e1c*ones(8,1));
        E1(1,2) = -e1s; E1(2,1) =  e1s; E1(3,4) =  e1s; E1(4,3) = -e1s;
        E1(5,6) =  e1s; E1(6,5) = -e1s; E1(7,8) = -e1s; E1(8,7) =  e1s;
        E2 = diag(e2c*ones(8,1));
        E2(1,3) = -e2s; E2(2,4) = -e2s; E2(3,1) =  e2s; E2(4,2) =  e2s;
        E2(5,7) =  e2s; E2(6,8) =  e2s; E2(7,5) = -e2s; E2(8,6) = -e2s;
        E4 = diag(e4c*ones(8,1));
        E4(1,5) = -e4s; E4(2,6) = -e4s; E4(3,7) = -e4s; E4(4,8) = -e4s;
        E4(5,1) =  e4s; E4(6,2) =  e4s; E4(7,3) =  e4s; E4(8,4) =  e4s;
        y = y + E1 * (E2 * (E4 * x));
    end; end; end
    y = y/(N1*N2*N3);
    Y(k1+1,k2+1,k3+1) = octonion(y(1),y(2),y(3),y(4),y(5),y(6),y(7),y(8));
end; end; end
```

Listing 6 alpha.m

```
function O = alpha(o,i1,i2,i3,i4)
% ALPHA Implements the composition of four functions alpha_i(o) =
% -e_i*(o*e_i), where i are indices given as ALPHA function arguments.
%
% This implementation uses the fact -ALPHA(o,i1,i2,i3,i4) changes the sign
% of four (different) imaginary units of an octonion o.

narginchk(5, 5), nargoutchk(0, 1)

if ~isreal(o)
    error('The first argument must be a real octonion.');
end

in = [i1, i2, i3, i4];
if numel(in)~=numel(unique(in))
    error('Indices i1, i2, i3 and i4 are not unique.');
end

if ~all(mod(in,1) == 0) || ~all(in >= 1) || ~all(in <= 7)
```

```
    error('Indices i1, i2, i3 and i4 must be values from set {1,...,7}.');
end

o0 = part(o,1); o4 = part(o,5);
o1 = part(o,2); o5 = part(o,6);
o2 = part(o,3); o6 = part(o,7);
o3 = part(o,4); o7 = part(o,8);

for i = in
    switch i
        case 1, o1 = -o1;
        case 2, o2 = -o2;
        case 3, o3 = -o3;
        case 4, o4 = -o4;
        case 5, o5 = -o5;
        case 6, o6 = -o6;
        case 7, o7 = -o7;
    end
end

O = octonion(-o0,-o1,-o2,-o3,-o4,-o5,-o6,-o7);
```

Listing 7 tutorial.m

```
%% OCTONION FOURIER TRANSFORM tutorial
%%
% Functions offt3 and iofft3 implement the octonion Fourier transform
% (forward and inverse). The qtfm (http://qtfm.sourceforge.net) package is
% required in order to use those functions. The argument of offt3 and
% iofft3 must be of an octonion type.

% We create sample 3-dimensional octonion matrix:
N = 4;
u = octonion(...
    rand(N,N,N), rand(N,N,N), rand(N,N,N), rand(N,N,N), ...
    rand(N,N,N), rand(N,N,N), rand(N,N,N), rand(N,N,N));

% And calculate its OFT with two different algorithms:
U1 = offt3(u);              % FFT-based implementation
U2 = doft3(u);              % direct formula-based implementation

%%
% We check the correctness of implemented functions by comparing outputs of
% two different algorithms:
dif = zeros(1,8);
for i = 1:8
    dif(i) = max(abs(part(U1(:)-U2(:),i)./part(U2(:),i)));
end
disp(max(dif)*100)          % relative error of OFFT3 function

%%
% The other way to check the correctness of FFT-based algorithm (by
% calculating the inverse transform):
v1 = iofft3(U1);
err1 = u-v1;
dif = zeros(1,8);
for i = 1:8
    dif(i) = max(abs(part(err1(:),i)./part(u(:),i)));
end
disp(max(dif)*100)          % relative error of reconstruction

%%
% Checking the correctness of direct formula-based algorithm (by
% calculating the inverse transform):
v2 = idoft3(U2);
err = u-v2;
```

```
dif = zeros(1,8);
for i = 1:8
    dif(i) = max(abs(part(err(:),i)./part(u(:),i)));
end
disp(max(dif)*100)          % relative error of reconstruction

%%
% We check if the OFFT has the symmetry properties stated in literature.
% We create sample 3-dimensional octonion matrix (with real values):
N = 32;
u = dc(dc(randi(9,N,N,N), zeros(N,N,N)), zeros(N,N,N));

% And calculate its OFT with FFT-based algorithm:
U = offt3(u);

% We check the symmetry with respect to 1st, 2nd and 3rd variable
dif1 = fftreflection(U,1) + alpha(U,1,3,5,7);
dif2 = fftreflection(U,2) + alpha(U,2,3,6,7);
dif3 = fftreflection(U,3) + alpha(U,4,5,6,7);
                            % absolute difference
disp([max(abs(dif1(:))), max(abs(dif2(:))), max(abs(dif3(:)))])
```

References

Baez JC (2002) The octonions. Bull Am Math Soc 39:145–205. https://doi.org/10.1090/S0273-0979-01-00934-X

Bahri M, Surahman (2013) Discrete quaternion fourier transform and properties. Int J Math Anal 7(25):1207–1215

Błaszczyk Ł (2018) Octonion spectrum of 3d octonion-valued signals—properties and possible applications. In: Proceedings of 2018 26th European signal processing conference (EUSIPCO), pp 509–513. https://doi.org/10.23919/EUSIPCO.2018.8553228

Błaszczyk Ł (2019) Hypercomplex Fourier transforms in the analysis of multidimensional linear time-invariant systems. In: Progress in industrial mathematics at ECMI 2018, pp 575–581. Springer Nature Switzerland AG. https://doi.org/10.1007/978-3-030-27550-1_73

Błaszczyk Ł (2020) A generalization of the octonion Fourier transform to 3-d octonion-valued signals—properties and possible applications to 3-d lti partial differential systems. Multidim Syst Sign Process 31(4):1227–1257. https://doi.org/10.1007/s11045-020-00706-3

Błaszczyk Ł, Snopek KM (2017) Octonion Fourier transform of real-valued functions of three variables—selected properties and examples. Signal Process 136:29–37. https://doi.org/10.1016/j.sigpro.2016.11.021

Brackx F, Hitzer E, Sangwine SJ (2013) History of quaternion and clifford Fourier transforms and wavelets. In: Hitzer E, S. Sangwine (eds.) Quaternion and Clifford Fourier Transforms and Wavelets, Trends in Mathematics, vol. 27, pp. xi–xxvii. Springer Basel AG. https://doi.org/10.1007/978-3-0348-0603-9

Delsuc MA (1988) Spectral representation of 2d nmr spectra by hypercomplex numbers. J Magn Reson 77:119–124

Ell TA (1993) Quaternion-fourier transforms for analysis of 2-dimensional linear time-invariant partial-differential systems. In: Proceedings of 32nd IEEE conference on decision and Controll, vol 1–4, pp 1830–1841

Ell TA, Bihan NL, Sangwine SJ (2014) Quaternion Fourier transforms for signal and image processing. Wiley-ISTE

Felsberg M, Bülow T, Sommer G (2001) Commutative hypercomplex Fourier transforms of multidimensional signals. In: Sommer G (ed) Geometric computing with clifford algebras. Theoretical foundations and applications in computer vision and robotics, pp 209–229. Springer, Berlin. https://doi.org/10.1007/978-3-662-04621-0_8

Gao HY, Lam KM (2014) From quaternion to octonion: feature-based image saliency detection. In: 2014 IEEE International conference on acoustics, speech and signal processing (ICASSP), pp 2808–2812

Gomes N, Hartmann S, Kähler U (2017) Compressed sensing for quaternionic signals. Complex Anal Oper Theory 11:417–455

Grigoryan AM, Agaian SS (2018) Quaternion and octonion color image processing with MATLAB. SPIE

Hahn SL, Snopek KM (2016) Complex and hypercomplex analytic signals: theory and applications. Artech House

Klco P, Smetana M, Kollarik M, Tatar M (2017) Application of octonions in the cough sounds classification. Adv Appl Sci Res 8(2):30–37

Kurman K (1958) Liczby podwójne zespolone i możliwość ich zastosowania. Tech. rep, Politechnika Warszawska, Katedra Automatyki i Telemechaniki

Lazendić S, Bie HD, Pižurica A (2018a) Octonion sparse representation for color and multispectral image processing. In: Proceeding 2018 26th European signal processing conference (EUSIPCO), pp 608–612

Lazendić S, Pižurica A, Bie HD (2018b) Hypercomplex algebras for dictionary learning. In: Proceedings the 7th conference on applied geometric algebras in computer science and engineering–AGACSE 2018, pp 57–64

Lian P (2019) The octonionic fourier transform: Uncertainty relations and convolution. Sig Process 164:295–300. https://doi.org/10.1016/j.sigpro.2019.06.015

Popa CA (2016) Octonion-valued neural networks. Artif Neural Netw Mach Learn ICANN 2016:435–443

Popa CA (2018) Global exponential stability of octonion-valued neural networks with leakage delay and mixed delays. Neural Netw 105:277–293

Sangwine S, Bihan NL (2005–2019) Quaternion and octonion toolbox for matlab. http://qtfm.sourceforge.net/

Sangwine SJ (1997) The discrete quaternion Fourier transform. In: 1997 6th International conference on image processing and its applications, pp 790–793. https://doi.org/10.1049/cp:19971004

Sheng H, Shen X, Lyu Y, Shi Z (2018) Image splicing detection based on Markov features in discrete octonion cosine transform domain. IET Image Proc 12(10):1815–1823

Snopek KM (2015) Quaternions and octonions in signal processing—fundamentals and some new results. Telecommunication review + telecommunication news, tele-radio-electronic, Information Technology 6:618–622

Tian Y (2000) Matrix representations of octonions and their applications. Adv Appl Clifford Algebras 10(1):61–90. https://doi.org/10.1007/BF03042010

Wang R, Xiang G, Zhang F (2017) L1-norm minimization for octonion signals. In: 2016 International conference on audio, language and image processing (ICALIP), pp 552–556

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.