Piotr Porwik

Wybrane metody cyfrowego przetwarzania sygnałów z przykładami programów w Matlabie

WYDAWNICTWO
UNIWERSYTETU ŚLĄSKIEGO
KATOWICE 2015

Wybrane metody cyfrowego przetwarzania sygnałów z przykładami programów w Matlabie

NR 3309

Piotr Porwik

Wybrane metody cyfrowego przetwarzania sygnałów z przykładami programów w Matlabie

Wydawnictwo Uniwersytetu Śląskiego • Katowice 2015

Redaktor serii: Informatyka i Inżynieria Biomedyczna
Mariusz Boryczka

Recenzent
Michał Woźniak

Spis treści

1. Przedmowa

Próby zastępowania sygnału analogowego sygnałem cyfrowym są podejmowane od wielu lat. Zalety sygnału cyfrowego są niezaprzeczalne, to powoduje, że w ostatnich latach wprost lawinowo rośnie liczba urządzeń przetwarzających sygnał w sposób cyfrowy. Obecnie jest to już konstatacja trywialna. Sygnałem może być ciąg próbek wyodrębnionych z sygnału ciągłego lub inny dyskretny zbiór danych – na przykład komputerowy obraz w formacie bitmapy. Należy jednak sobie uświadomić, że dane cyfrowe są ściśle powiązane z sygnałami analogowymi, które przecież powszechnie występują w przyrodzie. Dane cyfrowe są przeważnie odpowiednio pobieranymi w procesie próbkowania chwilowymi wartościami sygnału analogowego. Dane tego typu można zapisywać w postaci wektorów, gdzie współrzędna wektora stanowi pojedynczą wartość próbki. Dziedziną i zbiorem wartości każdego sygnału cyfrowego są wartości dyskretne, a te można modelować w przestrzeniach wektorowych. Metody korzystające z pojęć przestrzeni wektorowej, podobnie jak metody przestrzeni funkcyjnych, umożliwiają reprezentację danych za pomocą kombinacji liniowej wektorów bazowych. Dobór odpowiedniego zbioru wektorów bazowych (bazy) może być dokonywany różnie. Taki sposób opisu sygnału pierwotnego oznacza zawsze jego reprezentację za pomocą skończonego zbioru współczynników odpowiadającej mu kombinacji liniowej wektorów bazy. Współczynniki te nazwane są również współczynnikami widmowymi względem rozpatrywanych funkcji bazowych. Ich uporządkowany zbiór jednoznacznie reprezentuje sygnał pierwotny. Analiza tych współczynników – ich wartości i miejsca wystąpienia – pozwala na odkrywanie cech sygnału, co może być utrudnione lub niemożliwe w bezpośredniej obserwacji danych pierwotnych. Wymienione zagadnienia są między innymi tematem niniejszej monografii. Chociaż teoria i praktyka przetwarzania analogowego i cyfrowego wzajemnie się przenikają, o czym będzie mowa w dalszej części pracy, to przedstawione zagadnienia dotyczą cyfrowej analizy danych, z uwzględnieniem praktycznej wiedzy wynikającej z informacji o rozkładzie współczynników widmowych.

Niniejsza książka jest adresowana do tych Czytelników, którzy zainteresowani są metodami analizy sygnałów cyfrowych. Pierwsza część monografii ma charakter teoretyczny, omówiono w niej wybrane sposoby transformacji sygnałów dyskretnych w różnych bazach, w których funkcjami bazowymi mogą być zarówno funkcje trygonometryczne, jak również funkcje odcinkowo-stałe o odpowiednich własnościach. Rozważania teoretyczne znajdują wiele praktycznych zastosowań. Firmy produkujące zintegrowane systemy obliczeniowe zainteresowane są przyśpieszaniem pracy komputerów. Nowe rekonfigurowalne architektury sprzętowe oparte na układach FPGA (ang. *Field Programmable Gate Arrays*) pozwalają na wręcz skokowe przyspieszenie obliczeń, gdyż wiele opisanych w tej książce algorytmów można realizować sprzętowo. Omówione techniki obserwacji sygnałów znajdują zastosowanie w analizie dźwięku, systemach wizyjnych, przetwarzaniu obrazów, filtracji cyfrowej i wielu innych.

Rozważania teoretyczne objaśniane są za pomocą przykładów rachunkowych i kompletnych programów komputerowych, realizujących wybrane algorytmy. Wywody teoretyczne pozwalają bardziej dociekliwemu Czytelnikowi na śledzenie przekształceń matematycznych, które w efekcie końcowym umożliwiają konstrukcję odpowiednich algorytmów i ich zapis w języku programowania.

Dla realizacji programów komputerowych wykorzystano znane środowisko programistyczne Matlab, przeznaczone do zapisu algorytmów, wizualizacji, analizy danych oraz obliczeń numerycznych. W rozwiązaniach programowych zastosowano środowisko Matlab ver. 7.0, ale programy mogą być również uruchamiane w najnowszych wersjach programu Matlab. Można również korzystać z odpowiedników Matlaba, udostępnianych na licencji FLOSS (ang. *Free Open Source Software*), takich jak Octave czy Scilab.

Od Czytelnika wymaga się jedynie podstawowych umiejętności programowania, gdyż Matlab dostarcza wiele gotowych funkcji i procedur wywoływanych pojedynczym poleceniem. Czytelnikom nieznającym programu Matlab można polecić wiele prac opisujących wyczerpująco to środowisko programistyczne od strony formalnej i praktycznej [10, 22, 27, 39].

W drugiej części monografii pokazano, w jaki sposób wiedzę teoretyczną zastosować do rozwiązywania niektórych zadań inżynierskich. Tym zagadnieniom poświęcony został rozdział ostatni, w którym pokazano zastosowania transformacji jedno- i dwuwymiarowych w wydobywaniu kierunkowych szczegółów obrazu rzeczywistego. W tym samym rozdziale przedstawiono również sposoby analizy binarnych funkcji boolowskich metodami widmowymi oraz sposoby rozszerzania tego typu funkcji do form pełnych.

Część prezentowanych tutaj materiałów była wykorzystywana w trakcie prowadzonych przeze mnie wykładów dla studentów studiów informatycznych w Instytucie Informatyki, na Wydziale Informatyki i Nauki o Materiałach Uniwersytetu Śląskiego. Zatem książka ta może służyć także jako podręcznik.

Uważny Czytelnik dostrzeże, że prezentowana w niniejszej monografii tematyka została omówiona w sposób szczegółowy, a ujęcie poszczególnych partii materiału nie wymaga studiowania literatury dodatkowej. Autor ma nadzieję, że zdołał zrealizować przyjęte założenie, by każdy z rozdziałów stanowił zamkniętą całość, którą można studiować niezależnie. Przyjęcie takiej formuły pozwoli Czytelnikowi skupić się na poznawaniu opisywanych w kolejnych rozdziałach zagadnień, bez konieczności poszukiwania literatury uzupełniającej. Niezbędne pozycje literaturowe są oczywiście w odpowiednich miejscach przywoływane, pozwalając dociekliwemu odbiorcy na lekturę dodatkowych prac. Załączona na końcu książki bibliografia ma zarówno wskazać prace, na podstawie których dokonywano wywodów teoretycznych, jak i skłonić Czytelnika do dalszej lektury.

W tym miejscu chcę podziękować dr. hab. Michałowi Baczyńskiemu z Instytutu Matematyki Uniwersytetu Śląskiego w Katowicach za dyskusje dotyczące różnych zagadnień opisywanych w tej książce.

Niniejsza praca przybrała ostateczny kształt dzięki szczegółowym uwagom Recenzenta prof. dr. hab. inż. Michała Woźniaka z Politechniki Wrocławskiej, za które jestem Mu niezmiernie wdzięczny.

Książkę pragnę zadedykować Żonie Teresie oraz moim Córkom Monice, Ewie i Marcie. To One wspierały mnie w pracy nad niniejszą monografią. Jestem Im za to niezmiernie wdzięczny i wiem, że nie oddadzą tego żadne słowa. Tuż przed ukończeniem pracy nad książką Monika urodziła synka Adasia. Po raz pierwszy zostałem dziadkiem. To sprawiło, że postanowiłem szybko, z oczywistych względów, ukończyć monografię. Tak więc również Adaś przyczynił się do jej wydania.

2. Algebra liniowa. Pojęcia podstawowe

Zaprezentowane w niniejszym rozdziale pojęcia nie opisują w całości zagadnień algebry liniowej i zostały przedstawione wybiórczo, w możliwie prosty, zwarty sposób. Pozwalają jednak Czytelnikowi ze zrozumieniem studiować materiały zawarte w kolejnych rozdziałach książki. Przytoczone w tym rozdziale informacje podane są bez dowodów, gdyż te można znaleźć w bogatej literaturze przedmiotu. Osoby zainteresowane dokładnym poznaniem wszystkich działów algebry liniowej powinny zapoznać się ze źródłowymi pracami poświęconymi tym zagadnieniom [16,21]. Prawdziwość przedstawionych w niniejszym rozdziale zależności można sprawdzić samodzielnie, wykorzystując do tego celu odpowiednie funkcje zawarte w takich dedykowanych programach matematycznych, jak: Mathematica, Matlab czy Maple. Programy tego typu są nieustannie rozwijane i dostępne w wersjach pełnych (profesjonalnych) bądź "okrojonych" – studenckich, w których ogranicza się np. ich funkcjonalność lub wielkość danych, na których można przeprowadzać obliczenia.

2.1. Macierze

Układ liczb $m \times n$, gdzie $m, n \in \boldsymbol{N}$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix} = \boldsymbol{A}, \tag{2.1}$$

rozmieszczonych w m wierszach i n kolumnach nazywamy macierzą. Macierz może być traktowana również jako tablica. Element macierzy $\boldsymbol{A}$ stojący w i-tym wierszu oraz w j-tej kolumnie oznaczany jest symbolem a_{ij}. Macierz $\boldsymbol{A}$ można także przedstawiać w innym zapisie, w postaci $[a_{ij}]_{m\times n}$ lub $[a_{ij}]$, gdy

znany jest wymiar analizowanej macierzy. Macierz jest więc funkcją określoną w zbiorze par liczb $\{i, j\}$, gdzie $i = 1, ..., m, \quad j = 1, ..., n$, która parze $\{i, j\}$ przyporządkowuje element a_{ij}.

2.1.1. Działania ma macierzach

Dodawanie macierzy

Niech $\boldsymbol{A} = [a_{ij}]_{m\times n}$ oraz niech $\boldsymbol{B} = [b_{ij}]_{m\times n}$. Warunkiem koniecznym i dostatecznym dodawania (odejmowania) macierzy jest zgodność ich wymiarów. Sumą (różnicą) dwóch macierzy $\boldsymbol{A}$ i $\boldsymbol{B}$ jest macierz $\boldsymbol{C} = [c_{ij}]_{m\times n}$, której elementy określone są wzorem:

$$c_{ij} = a_{ij} \pm b_{ij}, \tag{2.2}$$

dla $i = 1, ..., m$ oraz $j = 1, ..., n$. Wtedy $\boldsymbol{C}$=$\boldsymbol{A} \pm \boldsymbol{B}$.

Mnożenie macierzy przez liczbę

Niech $\boldsymbol{A} = [a_{ij}]_{m\times n}$ oraz λ jest dowolną liczbą, wtedy:

$$\lambda\boldsymbol{A} = \boldsymbol{A}\lambda = [\lambda a_{ij}]. \tag{2.3}$$

Iloczyn macierzy

Niech $\boldsymbol{A} = [a_{ij}]_{m\times n}$ oraz $\boldsymbol{B} = [b_{ij}]_{n\times q}$. Iloczynem dwóch macierzy $\boldsymbol{A}$ oraz $\boldsymbol{B}$ nazywamy macierz $\boldsymbol{C} = [c_{ij}]_{m\times q}$, której elementy wyznaczane są według zależności:

$$c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{in}b_{nj}, \tag{2.4}$$

dla $i = 1, ..., m$ oraz $j = 1, ..., q$.

Iloczyn macierzy $\boldsymbol{C} = \boldsymbol{AB}$ można obliczyć tylko wtedy, gdy liczba kolumn macierzy $\boldsymbol{A}$ jest równa liczbie wierszy macierzy $\boldsymbol{B}$. Mnożenie macierzy, poza przypadkami szczególnymi, nie jest przemienne, co oznacza, że $\boldsymbol{AB} \neq \boldsymbol{BA}$.

Definicja 2.1. *Dana jest macierz* $\boldsymbol{A}$ *o wymiarach* $m \times n$ *z elementami* a_{ij} *oraz macierz* $\boldsymbol{B}$ *o wymiarach* $r \times s$ *z elementami* b_{ij}*. Iloczynem Kroneckera macierzy* $\boldsymbol{A}$ *oraz* $\boldsymbol{B}$ *jest macierz* $\boldsymbol{D}$ *o wymiarach* $mr \times ns$ *zbudowana w sposób następujący*:

$$\boldsymbol{D} = \boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \dots & a_{2n}\boldsymbol{B} \\ \dots & \dots & \dots & \dots \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix}, \tag{2.5}$$

gdzie symbol $\otimes$ *oznacza mnożenie macierzy* (*tzw. kronekerowskie*) *według reguły* (2.5).

2.1.2. Rodzaje macierzy

1. Macierz $\boldsymbol{A} = [a_{ij}]_{m \times n}$, w której wszystkie elemeny są równe 0, nazywamy **macierzą zerową**. Inaczej mówiąc, jeśli $\boldsymbol{A} = [a_{ij}]$ jest macierzą zerową, to wszystkie elementy $a_{ij} = 0$, dla $i = 1, ..., m$, $j = 1, ..., n$. Taką macierz oznacza się często symbolem $\mathbf{0}_{m \times n}$. Należy również zauważyć, że z równania $\boldsymbol{AB} = \mathbf{0}$ nie wynika, że $\boldsymbol{A} = \mathbf{0}$ lub $\boldsymbol{B} = \mathbf{0}$, gdyż istnieją niezerowe macierze $\boldsymbol{A}$ ($\boldsymbol{A} \neq \mathbf{0}$) lub $\boldsymbol{B}$ ($\boldsymbol{B} \neq \mathbf{0}$), których iloczyn jest macierzą zerową [36].

2. **Macierz kwadratowa** $\boldsymbol{A}$ jest macierzą o wymiarze $n \times n$. Liczbę wierszy (kolumn) takiej macierzy nazywamy stopniem macierzy. Te elementy a_{ij} macierzy $\boldsymbol{A}$, które posiadają oba wskaźniki równe, tworzą w macierzy kwadratowej przekątną główną.

3. Macierz $\boldsymbol{A} = [a_{ij}]_{n \times n}$ jest **macierzą osobliwą**, gdy:

$$\det(\boldsymbol{A}) = 0, \tag{2.6}$$

   w przeciwnym przypadku, gdy $det(\boldsymbol{A}) \neq 0$, macierz $\boldsymbol{A}$ jest **nieosobliwa**.

4. Macierz $\boldsymbol{A} = [a_{ij}]_{n \times n}$, w której wszystkie elementy niezerowe znajdują się na głównej przekątnej, nazywa się **macierzą diagonalną**.

Dla macierzy diagonalnych spełnione są następujące warunki:

$$a_{ij} = \begin{cases} d_i & \text{dla} \quad i = j \\ 0 & \text{dla} \quad i \neq j \end{cases} . \tag{2.7}$$

Macierz diagonalna oznaczana jest również symbolem $diag(d_1, d_2, ..., d_n)$, gdzie $d_i = a_{ii}$ są elementami głównej przekątnej macierzy. Jeśli $d_i = 1$, macierz $\boldsymbol{A}$ nazywa się **macierzą jednostkową**. Macierz jednostkową stopnia n zwykle oznacza się symbolem $\boldsymbol{I}_n$. Jeśli $d_i = \lambda_i$, diagonalna macierz $\boldsymbol{A}$ nazywa się **macierzą skalarną**. Jeśli $\boldsymbol{A}$ jest macierzą skalarną, to $diag(\lambda_1, \lambda_2, ..., \lambda_n) = \lambda \boldsymbol{I}_n$. Zauważmy również, że $\lambda \boldsymbol{I}_n \boldsymbol{A} = \lambda \boldsymbol{A}$. Szczególnym przypadkiem macierzy skalarnej jest macierz jednostkowa. Macierze jednostkowe i skalarne są przemienne z dowolną macierzą tego samego stopnia, co oznacza, że $\boldsymbol{A}\boldsymbol{I}_n = \boldsymbol{I}_n \boldsymbol{A}$.

5. Dla macierzy $\boldsymbol{A} = [a_{ij}]_{m \times n}$ **macierzą transponowaną** jest macierz $\boldsymbol{B} = [b_{ij}]_{m \times n}$, której elementy określone są wzorem:

$$b_{ij} = a_{ji}, \tag{2.8}$$

dla $i = 1, ..., m$ oraz $j = 1, ..., n$. Macierz transponowaną do macierzy $\boldsymbol{A}$ oznacza się symbolem $\boldsymbol{A}^T$.

6. Jeżeli $\boldsymbol{A}^T = \boldsymbol{A}$, to macierz $\boldsymbol{A}$ nazywa się **macierzą symetryczną**. Jeśli $\boldsymbol{A}^T = -\boldsymbol{A}$, to macierz $\boldsymbol{A}$ nazywa się **macierzą antysymetryczną** albo **skośnosymetryczną**.

7. Niech $\boldsymbol{A} = [a_{ij}]_{n \times n}$. Macierz $\boldsymbol{A}$ jest **macierzą odwracalną**, jeśli istnieje taka macierz $\boldsymbol{B} = [b_{ij}]_{n \times n}$, że zachodzi:

$$\boldsymbol{A} \cdot \boldsymbol{B} = \boldsymbol{B} \cdot \boldsymbol{A} = \boldsymbol{I}_n. \tag{2.9}$$

Wyznacznik macierzy odwracalnej jest różny od zera: $det(\boldsymbol{A}) \neq 0$.

8. **Macierzą odwrotną** do macierzy $\boldsymbol{A} = [a_{ij}]_{n\times n}$ jest macierz kwadratowa oznaczana symbolem $\boldsymbol{A}^{-1}$, która spełnia warunek:

$$\boldsymbol{A} \cdot \boldsymbol{A}^{-1} = \boldsymbol{A}^{-1} \cdot \boldsymbol{A} = \boldsymbol{I}_n, \tag{2.10}$$

$$\boldsymbol{A}^{-1}\boldsymbol{I}_n = \boldsymbol{A}^{-1}. \tag{2.11}$$

Jeżeli macierz $\boldsymbol{A}$ posiada macierz odwrotną, to $\boldsymbol{A}$ jest macierzą odwracalną.

9. Niech $\boldsymbol{A} = [a_{ij}]_{n\times n}$. Macierz $\boldsymbol{A}$ jest **macierzą ortogonalną**, jeśli spełniony jest warunek:

$$\boldsymbol{A}^{-1} = \boldsymbol{A}^{T}. \tag{2.12}$$

Z warunków (2.10) oraz (2.12) wynika, że dla macierzy ortogonalnych spełnione jest także równanie:

$$\boldsymbol{A} \cdot \boldsymbol{A}^{T} = \boldsymbol{I}_n. \tag{2.13}$$

Dla macierzy ortogonalnej $det(\boldsymbol{A}) = 1$. Jeśli macierz jest macierzą ortogonalną, to wektory wierszowe tej macierzy tworzą bazę ortonormalną przestrzeni $\boldsymbol{E}^n$. Bazę ortonormalną przestrzeni $\boldsymbol{E}^n$ tworzą również wektory kolumnowe macierzy $\boldsymbol{A}$.

10. Niech K będzie ciałem zawartym w ciele liczb zespolonych Z. Jeśli dana jest liczba $x = a+bi$, to przez $\overline{x}$ oznacza się liczbę sprzężoną $a-bi$. **Macierzą sprzężoną trywialnie** z macierzą $\boldsymbol{A} = [a_{ij}]_{m\times n}$ nazywamy macierz oznaczaną symbolem $\overline{\boldsymbol{A}}$, której każdy element jest liczbą sprzężoną do odpowiadającego mu elementu macierzy $\boldsymbol{A}$:

$$a_{ij} \mapsto \overline{a}_{ij}. \tag{2.14}$$

Jeśli więc $\boldsymbol{A} = \begin{bmatrix} 3 & 1+i \\ 2i & 20-4i \end{bmatrix}$, to macierzą trywialnie z nią sprzężoną jest macierz $\overline{\boldsymbol{A}} = \begin{bmatrix} 3 & 1-i \\ -2i & 20+4i \end{bmatrix}$.

11. **Macierzą sprzężoną hermitowsko** z macierzą $\boldsymbol{A}$ nazywamy macierz oznaczaną symbolem $\boldsymbol{A}^*$, dla której spełniona jest zależność:

$$\boldsymbol{A}^* = \overline{\boldsymbol{A}}^T. \tag{2.15}$$

12. **Macierz hermitowska** $\boldsymbol{A}$ jest macierzą, dla której zachodzi:

$$\boldsymbol{A} = \boldsymbol{A}^*. \tag{2.16}$$

Inaczej mówiąc, macierz hermitowska jest taką macierzą, która jest równa transpozycji swojej macierzy sprzężonej.

Jeśli więc $\boldsymbol{A} = \begin{bmatrix} 2 & 1+i \\ 1-i & -2 \end{bmatrix}$, to $\overline{\boldsymbol{A}} = \begin{bmatrix} 2 & 1-i \\ 1+i & -2 \end{bmatrix}$

oraz $\boldsymbol{A}^* = \overline{\boldsymbol{A}}^T = \begin{bmatrix} 2 & 1+i \\ 1-i & -2 \end{bmatrix} = \boldsymbol{A}$.

13. **Macierzą unitarną** $\boldsymbol{A} = [a_{ij}]_{n\times n}$ nazywamy macierz kwadratową spełniającą własność:

$$\boldsymbol{A}^* \cdot \boldsymbol{A} = \boldsymbol{I}_n, \tag{2.17}$$

co oznacza, że macierz unitarna $\boldsymbol{A}$ posiada macierz odwrotną $\boldsymbol{A}^*$, czyli $\boldsymbol{A}^* = \boldsymbol{A}^{-1}$. Tak więc macierz unitarna posiada macierz odwrotną będącą hermitowskim sprzężeniem jej samej.

2.2. Przestrzeń liniowa

Przestrzeń liniowa nazywana jest również przestrzenią wektorową i jest jednym z elementów algebry liniowej oraz analizy funkcjonalnej. Przestrzenie tego typu znajdują zastosowanie w bardzo wielu dziedzinach współczesnych badań naukowych. Naturalnymi przykładami przestrzeni liniowych są dwu- i trójwymiarowe przestrzenie euklidesowe. Przestrzeń liniowa jest strukturą algebraiczną, składającą się ze zbioru wektorów $\boldsymbol{V}$, ciała liczbowego K oraz działań dodawania i mnożenia wektorów.

Definicja 2.2. *Przestrzenią liniową nad ciałem K nazywamy niepusty zbiór $\boldsymbol{V}$ z dwoma działaniami dwuargumentowymi:*
– *dodawaniem wektorów: $\boldsymbol{a}+\boldsymbol{b}$, dla dowolnych $\boldsymbol{a},\boldsymbol{b}\in\boldsymbol{V}$,*
– *mnożeniem wektora przez skalar: $\alpha\boldsymbol{a}$, gdzie $\alpha\in K$ oraz $\boldsymbol{a}\in\boldsymbol{V}$.*

Dla wymienionych działań spełnione być muszą poniższe aksjomaty:

1. *Występuje przemienność dodawania: $\boldsymbol{a}+\boldsymbol{b}=\boldsymbol{b}+\boldsymbol{a}$.*

2. *Zapewniona jest łączność dodawania: $(\boldsymbol{a}+\boldsymbol{b})+\boldsymbol{c}=\boldsymbol{a}+(\boldsymbol{b}+\boldsymbol{c})$.*

3. *Istnieje element neutralny $\mathbf{o}$ przestrzeni $\mathbf{V}$ taki, że: $\boldsymbol{a}+\boldsymbol{o}=\boldsymbol{a}$.*

4. *Istnieje element przeciwny do danego elementu:*

dla każdego wektora $\boldsymbol{a}\in\boldsymbol{V}$ istnieje wektor $-\boldsymbol{a}\in\boldsymbol{V}$ taki, że $\boldsymbol{a}+(-\boldsymbol{a})=\boldsymbol{o}$.

5. $\alpha(\beta\boldsymbol{a})=(\alpha\beta)\boldsymbol{a}$.

6. $(\alpha+\beta)\boldsymbol{a}=\alpha\boldsymbol{a}+\beta\boldsymbol{a}$ *oraz* $\alpha(\boldsymbol{a}+\boldsymbol{b})=\alpha\boldsymbol{a}+\alpha\boldsymbol{b}$.

Przykład 2.1. *Niech $n\in\boldsymbol{N}$ oraz niech $\boldsymbol{R}^n$ będzie zbiorem wektorów o n współrzędnych takich, że:* $\boldsymbol{R}^n=\{\boldsymbol{a}=[a_1,a_2,...,a_n]:a_i\in R \text{ dla } 1\le i\le n\}$,

$\forall_{\boldsymbol{a},\boldsymbol{b}\in\boldsymbol{R}^n}\ [a_1,a_2,...,a_n]+[b_1,b_2,...,b_n]=[a_1+b_1,a_2+b_2,...,a_n+b_n]$,

$\forall_{\boldsymbol{a}\in\boldsymbol{R}^n}\ \forall_{\alpha\in R}\ \ \alpha\boldsymbol{a}=[\alpha a_1,\alpha a_2,...,\alpha a_n]$. *Tak zdefiniowany zbiór $\boldsymbol{R}^n$ jest wtedy przestrzenią liniową nad ciałem R.*

Definicja 2.3. *Niech $\boldsymbol{V}$ będzie przestrzenią liniową nad R. Iloczynem skalarnym w przestrzeni $\boldsymbol{V}$ nazywamy funkcję, która każdej parze wektorów tej przestrzeni przypisuje liczbę rzeczywistą $\langle\boldsymbol{a},\boldsymbol{b}\rangle$. Funkcja ta spełnia następujące warunki:*

1. $\langle\boldsymbol{a},\boldsymbol{b}\rangle=\langle\boldsymbol{b},\boldsymbol{a}\rangle$.

2. $\langle\boldsymbol{a}+\boldsymbol{b},\boldsymbol{c}\rangle=\langle\boldsymbol{a},\boldsymbol{c}\rangle+\langle\boldsymbol{b},\boldsymbol{c}\rangle$ *dla dowolnych* $\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}\in\boldsymbol{V}$.

3. $\langle\alpha\boldsymbol{a},\boldsymbol{b}\rangle=\alpha\langle\boldsymbol{a},\boldsymbol{b}\rangle$ *dla dowolnych* $\boldsymbol{a},\boldsymbol{b}\in\boldsymbol{V}$ *oraz dla każdego* $\alpha\in R$.

4. $\langle \boldsymbol{a}, \boldsymbol{a} \rangle \geq 0$ *dla każdego* $\boldsymbol{a} \in \boldsymbol{V}$.

5. $\langle \boldsymbol{a}, \boldsymbol{a} \rangle = 0$ *wtedy i tylko wtedy, gdy* $\boldsymbol{a} = \boldsymbol{o}$.

Niech $\boldsymbol{a} = [a_1, a_2, ..., a_n]$ oraz $\boldsymbol{b} = [b_1, b_2, ..., b_n]$, gdzie $n \in \boldsymbol{N}$ i $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{V}$. Wtedy iloczyn skalarny w przestrzeni $\boldsymbol{V}$ wyznaczany jest według następującej formuły:

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = a_1 b_1 + a_2 b_2 + ... + a_n b_n = \sum_{i=1}^{n} a_i b_i.$$

Przestrzeń liniowa z wprowadzonym do niej iloczynem skalarnym nazywana jest także przestrzenią unitarną, w której możliwe jest definiowanie takich pojęć, jak kąt, długość wektora (czyli norma elementu przestrzeni) lub ortogonalności elementów. Przestrzenie unitarne, zupełne ze względu na metrykę generowaną przez normę (zależną od iloczynu skalarnego), są przestrzeniami metrycznymi i nazywane są także przestrzeniami Hilberta.

Definicja 2.4. *Niech* $\boldsymbol{V}$ *będzie przestrzenią liniową nad ciałem* K. *Wektory* $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ *tej przestrzeni, gdzie* $n \in \boldsymbol{N}$, *są liniowo niezależne wtedy i tylko wtedy, gdy z warunku* $\alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + ... + \alpha_n \boldsymbol{v}_n = \boldsymbol{o}$ *dla dowolnych współczynników* $\alpha_1, \alpha_2, ..., \alpha_n \in K$ *wynika, że jedynymi takimi współczynnikami są skalary o wartościach* $\alpha_1 = \alpha_2 = ... = \alpha_n = 0$.

Definicja 2.5. *Niech* $\boldsymbol{V}$ *będzie przestrzenią liniową nad ciałem* K. *Wektory* $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ *tej przestrzeni, gdzie* $n \in \boldsymbol{N}$, *są liniowo zależne wtedy i tylko wtedy, gdy* $\bigvee_{\alpha_i \neq 0}$ *(czyli nie wszystkie współczynniki równocześnie są zerami), dla których spełniona jest równość* $\alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + ... + \alpha_n \boldsymbol{v}_n = \boldsymbol{o}$. *Oznacza to, że wektor* $\boldsymbol{v}_i$ *może być wpisany w liniowy związek z innym.*

Definicja 2.6. *Układ* $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ *liniowo niezależnych wektorów przestrzeni nazywa się układem maksymalnym tej przestrzeni, jeżeli w wyniku dołączenia do tego układu dowolnego wektora* $\boldsymbol{v}_{n+1}$ *otrzymamy układ wektorów liniowo zależnych.*

Na podstawie wymienionych powyżej definicji można przedstawić ważne właściwości wektorów liniowo zależnych i liniowo niezależnych.

Niech $\boldsymbol{V}$ będzie przestrzenią liniową nad ciałem K oraz niech $\boldsymbol{a}$,$\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ będą wektorami tej przestrzeni. Niech $\boldsymbol{W}$ będzie podprzestrzenią liniową przestrzeni $\boldsymbol{V}$, zbudowaną nad tym samym ciałem K, wówczas [16,31]:

- wektor $\boldsymbol{a}$ jest liniowo niezależny wtedy i tylko wtedy, gdy $\boldsymbol{a} \neq \boldsymbol{o}$,
- wektory $\boldsymbol{o}, \boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ są liniowo zależne,
- jeżeli wektory $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ są liniowo zależne, to wektory $\boldsymbol{a}$,$\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ są również liniowo zależne,
- jeżeli wektory $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n \in \boldsymbol{W}$ są liniowo zależne (niezależne) w przestrzeni $\boldsymbol{V}$, to są również zależne (niezależne) w przestrzeni $\boldsymbol{W}$.

Definicja 2.7. *Niech $\boldsymbol{V}$ będzie przestrzenią liniową nad ciałem K. Niech $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n \in \boldsymbol{V}$ oraz $\alpha_1, \alpha_2, ..., \alpha_n \in K$, wtedy wektor $\boldsymbol{b} = \alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + ... + \alpha_n \boldsymbol{v}_n$ nazywa się kombinacją liniową wektorów.*

Warunkiem wystaczającym na to, aby wektory $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n \in \boldsymbol{V}$, dla $n \geq 2$, były liniowo zależne jest warunek, aby co najmniej jeden z nich był kombinacją liniową pozostałych. Gdy takiej kombinacji liniowej nie można znaleźć, wektory są niezależne. Opisywany warunek można również odnieść do przestrzeni nieskończenie wymiarowych. Nieskończony zbiór wektorów przestrzeni liniowej $\boldsymbol{V}^{\infty}$ jest liniowo niezależny, jeżeli każdy jego skończony podzbiór wektorów jest liniowo niezależny. Gdy tak nie jest, zbiór takich wektorów jest liniowo zależny.

2.2.1. Przestrzeń euklidesowa

Definicja 2.8. *Przestrzeń liniowa, w której został wprowadzony iloczyn skalarny, nazywana jest przestrzenią euklidesową $\boldsymbol{E}$.*

Przykład 2.2. *Przywołując definicje podane w poprzednim paragrafie, można łatwo wykazać zależność lub niezależność różnych wektorów:*

1. *W przestrzeni euklidesowej $\boldsymbol{R}^n$ wektory $\boldsymbol{e}_1 = [1, 0, ..., 0]$, $\boldsymbol{e}_2 = [0, 1, ..., 0]$, ..., $\boldsymbol{e}_n = [0, 0, ..., 1]$ są liniowo niezależne. Dodatkowo wektory te stanowią układ maksymalny tej przestrzeni.*

Ponieważ wektory $\boldsymbol{e}_i$ są liniowo niezależne, to można powiedzieć, że przestrzeń $\boldsymbol{R}^n$ jest przestrzenią generowaną przez te wektory. Oznacza to, że przestrzeń $\boldsymbol{R}^n$jest przestrzenią euklidesową $\boldsymbol{E}^n$.

2. *W przestrzeni euklidesowej $\boldsymbol{R}^3$ nad ciałem R wektory $\boldsymbol{a} = [2, 1, 0], \boldsymbol{b} = [1, 2, 1]$ oraz $\boldsymbol{c} = [1, -1, -1]$ są liniowo zależne, ponieważ warunek $\alpha_1\boldsymbol{a} + \alpha_2\boldsymbol{b} + \alpha_3\boldsymbol{c} = \boldsymbol{o} = [0, 0, 0]$ może być spełniony dla wartości skalarów $\alpha_1 = -1$, $\alpha_2 = \alpha_3 = 1$, $\alpha_i \in R$.*

3. *W przestrzeni euklidesowej $\boldsymbol{R}^3$ nad ciałem R wektory $\boldsymbol{a} = [0, 1, 0]$ oraz $\boldsymbol{b} = [1, 0, 1]$ są wektorami liniowo niezależnymi, ponieważ warunek $\alpha_1\boldsymbol{a} + \alpha_2\boldsymbol{b} = \boldsymbol{o}$ może być spełniony jedynie dla skalarów o wartościach $\alpha_1 = \alpha_2 = 0$, $\alpha_i \in R$.*

Definicja 2.9. *Niech $\boldsymbol{a}$ będzie dowolnym wektorem przestrzeni euklidesowej $\boldsymbol{E}$. Normą wektora zadaną przez iloczyn skalarny nazywamy liczbę*:

$$\|\boldsymbol{a}\| = \sqrt{\langle \boldsymbol{a}, \boldsymbol{a} \rangle}. \tag{2.18}$$

Norma wektora nazywana jest także długością wektora.

Iloczyn skalarny w przestrzeni $\boldsymbol{E}$ wyznacza więc normę euklidesową zadaną wzorem (2.18). Wektor $\boldsymbol{a}$ przestrzeni euklidesowej $\boldsymbol{E}$ jest unormowany wtedy, gdy $\|\boldsymbol{a}\| = 1$. Dowolny wektor $\boldsymbol{a} \in \boldsymbol{E}$ można normować. Operacja normowania generuje nowy, unormowany wektor $\boldsymbol{b} = \dfrac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$ przestrzeni, który jest współliniowy z wektorem $\boldsymbol{a}$.

Definicja 2.10. *Dwa wektory $\boldsymbol{a}$ i $\boldsymbol{b}$ przestrzeni euklidesowej $\boldsymbol{E}$ są ortogonalne, jeżeli ich iloczyn skalarny wynosi zero, to znaczy jest spełniona równość*: $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = 0$.

Dwa wektory ortonormalne są unormowanymi wektorami ortogonalnymi. Wektor zerowy $\boldsymbol{o}$ jest ortogonalny do każdego wektora. Ortogonalność wektorów można również sprawdzić inaczej. Dwa wektory $\boldsymbol{a}$ i $\boldsymbol{b}$ przestrzeni euklidesowej są ortogonalne wtedy i tylko wtedy, gdy spełniony jest warunek:

$\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 = \|\boldsymbol{a}+\boldsymbol{b}\|^2$. Dla przestrzeni $\boldsymbol{E}^2$ ortogonalność wektorów oznacza, że są one prostopadłe. Pojęcie ortogonalności wektorów można również odnieść do przestrzeni $\boldsymbol{E}^n$, dla dowolnego n. Ważną własnością normy (2.18) w przestrzeniach rzeczywistych jest reguła równoległoboku:

$$2 \cdot \|\boldsymbol{a}\|^2 + 2 \cdot \|\boldsymbol{b}\|^2 = \|\boldsymbol{a}+\boldsymbol{b}\|^2 + \|\boldsymbol{a}-\boldsymbol{b}\|^2 . \tag{2.19}$$

Oznacza to, że iloczyn skalarny można również wyznaczyć z formuły:

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \tfrac{1}{2}(\|\boldsymbol{a}+\boldsymbol{b}\|^2 - \|\boldsymbol{a}\|^2 - \|\boldsymbol{b}\|^2). \tag{2.20}$$

Zbiór wektorów przestrzeni euklidesowej $\boldsymbol{E}$ nazywamy układem ortogonalnym wtedy i tylko wtedy, gdy dowolna para wektorów z tego zbioru jest ortogonalna. Zbiór wektorów przestrzeni $\boldsymbol{E}$ jest układem ortonormalnym, gdy składa się z unormowanych wektorów parami ortogonalnych.

Oznacza to, że dla każdej pary wektorów $\boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbf{E}$ mamy:

$$\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = \begin{cases} C & \text{dla} \quad i = j \\ 0 & \text{dla} \quad i \neq j \end{cases} . \tag{2.21}$$

Gdy $C = 1$, wektory $\boldsymbol{v}_i, \boldsymbol{v}_j$ są unormowane. Gdy $C \neq 1$, wymienione wektory nie są unormowane.

2.2.2. Ortogonalizacja Grama–Schmidta

Dowolny zbiór liniowo niezależnych wektorów, czyli bazę $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n\}$ przestrzeni euklidesowej, można przekształcić w zbiór wektorów ortogonalnych, czyli bazę ortogonalną $\{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_n\}$ tej samej przestrzeni, ortogonalizując ją w procesie Grama–Schmidta [31]. W takim przypadku wektory bazy ortogonalnej są sukcesywnie budowane na podstawie formuły:

$$\begin{cases} \boldsymbol{w}_1 = \boldsymbol{v}_1 \\ \boldsymbol{w}_2 = \boldsymbol{v}_2 - \dfrac{\langle \boldsymbol{v}_2, \boldsymbol{w}_1 \rangle}{\|\boldsymbol{w}_1\|^2} \boldsymbol{w}_1 \\ \boldsymbol{w}_3 = \boldsymbol{v}_3 - \left[\dfrac{\langle \boldsymbol{v}_3, \boldsymbol{w}_1 \rangle}{\|\boldsymbol{w}_1\|^2} \boldsymbol{w}_1 + \dfrac{\langle \boldsymbol{v}_3, \boldsymbol{w}_2 \rangle}{\|\boldsymbol{w}_2\|^2} \boldsymbol{w}_2 \right] . \end{cases} \tag{2.22}$$

Z opisu (2.22) wynika, że podaną procedurę ortogonalizacji wektorów można zapisać w sposób rekurencyjny:

$$\boldsymbol{w}_n = \boldsymbol{v}_n - \left[\frac{\langle \boldsymbol{v}_n, \boldsymbol{w}_1 \rangle}{\|\boldsymbol{w}_1\|^2} \circ \boldsymbol{w}_1 + \frac{\langle \boldsymbol{v}_n, \boldsymbol{w}_2 \rangle}{\|\boldsymbol{w}_2\|^2} \circ \boldsymbol{w}_2 + ... + \frac{\langle \boldsymbol{v}_n, \boldsymbol{w}_{n-1} \rangle}{\|\boldsymbol{w}_{n-1}\|^2} \circ \boldsymbol{w}_{n-1} \right]. \tag{2.23}$$

Aby zbudować w ten sposób zbiór ortonormalny, każdy wektor należy dodatkowo podzielić przez jego normę, co już omówiono wcześniej.

Przykład 2.3. *Dane są dwa wektory* $\boldsymbol{v}_1 = [1, 0, -2]$, $\boldsymbol{v}_2 = [5, 1, 4]$. *Należy zortogonalizować te wektory. Stosując bezpośrednio formułę* (2.22), *otrzymujemy*:

$$\boldsymbol{w}_1 = \boldsymbol{v}_1 = [1, 0, -2],$$

$$\boldsymbol{w}_2 = \boldsymbol{v}_2 - \frac{\langle \boldsymbol{v}_2, \boldsymbol{w}_1 \rangle}{\|\boldsymbol{w}_1\|^2} \boldsymbol{w}_1 = [5, 1, 4] - \frac{[5, 1, 4] \circ [1, 0, -2]}{5} \circ [1, 0, -2] = [\tfrac{28}{5}, 1, \tfrac{14}{5}].$$

Wektory $\boldsymbol{v}_1$ *oraz* $\boldsymbol{v}_2$ *rzeczywiście nie były ortogonalne, bo ich iloczyn skalarny* $\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle \neq 0$. *Po ortogonalizacji wektory* $\boldsymbol{w}_1$ *i* $\boldsymbol{w}_2$ *są już ortogonalne*:

$$\langle \boldsymbol{w}_1, \boldsymbol{w}_2 \rangle = [1, 0, -2] \circ [\tfrac{28}{5}, 1, \tfrac{14}{5}] = 0.$$

2.2.3. Metryka przestrzeni

Metryka przestrzeni jest funkcją podającą sposób obliczania odległości pomiędzy dwoma punktami należącymi do danej przestrzeni. Metryka opisuje więc geometryczne właściwości zbioru. Sformułowanie ogólnych warunków, jakie powinna spełniać ta odległość, prowadzi do pojęcia przestrzeni metrycznej.

Definicja 2.11. *Zbiór* $\boldsymbol{X}$ *nazywamy przestrzenią metryczną, jeśli każdej parze elementów* $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \boldsymbol{X}$ *przyporządkowana jest nieujemna liczba* d *w taki sposób, że spełnione są następujące aksjomaty metryki*:

1. $d(\boldsymbol{a}, \boldsymbol{b}) = 0 \Leftrightarrow \boldsymbol{a} = \boldsymbol{b}$.

2. $d(\boldsymbol{a}, \boldsymbol{b}) = d(\boldsymbol{b}, \boldsymbol{a})$.

3. $d(\boldsymbol{a}, \boldsymbol{b}) + d(\boldsymbol{b}, \boldsymbol{c}) \geq d(\boldsymbol{a}, \boldsymbol{c})$.

Funkcję d nazywamy metryką, a wartość funkcji $d(\boldsymbol{a}, \boldsymbol{b})$ dla ustalonej pary elementów $(\boldsymbol{a}, \boldsymbol{b})$ nazywamy odległością pomiędzy punktami $\boldsymbol{a}$ i $\boldsymbol{b}$, przy czym "odległość" jest w pewnym uproszczeniu synonimem pojęcia "niepodobieństwo". Zgodnie z aksjomatem nr 1 podanym w Definicji 2.11, jeśli wektory są identyczne, to ich niepodobieństwo wynosi zero. W zbiorze $\boldsymbol{X}$ metrykę można definiować w różny sposób. Metryki są odpowiednio zdefiniowanymi odległościami pomiędzy wektorami. Jeśli funkcja, według której wyznacza się odległość pomiędzy wektorami, nie spełnia warunku 3 Definicji 2.11, to funkcja d nazywana jest miarą, np. odległość euklidesowa jest metryką, ale kwadrat odległości euklidesowej nie jest metryką tylko miarą, gdyż nie spełnia wszystkich podanych powyżej aksjomatów metryki. Poniżej przedstawiono kilka najbardziej popularnych miar i metryk.

Niech $\boldsymbol{a} = [a_1, a_2, ..., a_n]$ oraz $\boldsymbol{b} = [b_1, b_2, ..., b_n]$, gdzie $n \in \boldsymbol{N}$ i $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{R}^n$, wtedy:

- metryka euklidesowa: $d_E(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$,
- metryka Manhattan (taksówkowa): $d_M(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{n} |a_i - b_i|$,
- metryka Czebyszewa: $d_C(\boldsymbol{a}, \boldsymbol{b}) = \max\{|a_i - b_i|,\ i = 1, ..., n\}$,
- metryka Minkowskiego: $d_K(\boldsymbol{a}, \boldsymbol{b}) = \left(\sum_{i=1}^{n} |a_i - b_i|^k\right)^{\frac{1}{k}}$,
  - dla $k = 1$ jest równoważna metryce d_M,
  - dla $k = 2$ jest równoważna metryce d_E,
  - dla $k \to \infty$ jest równoważna nieważonej metryce d_C,
- metryka Dice'a: $d_D(\boldsymbol{a}, \boldsymbol{b}) = \dfrac{2\sum_{i=1}^{n} a_i b_i}{\sum_{i=1}^{n} a_i^2 + \sum_{i=1}^{n} b_i^2}$,
- metryka Jaccarda: $d_J(\boldsymbol{a}, \boldsymbol{b}) = \dfrac{\sum_{i=1}^{n} a_i b_i}{\sum_{i=1}^{n} a_i^2 + \sum_{i=1}^{n} b_i^2 - \sum_{i=1}^{n} a_i b_i}$,
- miara kosinusowa: $d_{\cos}(\boldsymbol{a}, \boldsymbol{b}) = \dfrac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}|| \cdot ||\boldsymbol{b}||} = \dfrac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}$,

która wyraża kosinus kąta między znormalizowanymi wektorami. Miarę kosinusową można przekształcić do metryki:
$d_{\cos}(\boldsymbol{a}, \boldsymbol{b}) = 1 - \dfrac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}|| \cdot ||\boldsymbol{b}||}$,
dla której spełnione są teraz wszystkie podane aksjomaty z Definicji 2.11,

– miara Tanimoto: $d_T(\boldsymbol{a}, \boldsymbol{b}) = \dfrac{\boldsymbol{a}^T \cdot \boldsymbol{b}}{||\boldsymbol{a}||^2 + ||\boldsymbol{b}||^2 - \boldsymbol{a}^T \cdot \boldsymbol{b}}$.

Podane powyżej współczynniki służą do wyznaczania odległości w przestrzeniach metrycznych. W niektórych dziedzinach, takich jak przetwarzanie obrazów lub klasyfikacja i rozpoznawanie wzorców, wymienione miary odległości nazywane są także współczynnikami podobieństwa obiektów. Parametry (cechy) obiektów reprezentowane są wtedy przez poszczególne współrzędne wektorów $\boldsymbol{a}$ oraz $\boldsymbol{b}$ opisujących te obiekty.

2.2.4. Baza i wymiar przestrzeni liniowej

Definicja 2.12. *Układ wektorów $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ nazywamy bazą skończoną przestrzeni liniowej $\boldsymbol{V}$, jeśli*:

– $\boldsymbol{v}_i \in \boldsymbol{V}$ *dla* $i = 1, 2, ..., n$,

– *wektory $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ są liniowo niezależne.*

Aby układ wektorów tworzył bazę, musi spełniać dwa warunki: każdy punkt przestrzeni daje się przedstawić jako kombinacja wektorów z tego układu oraz można tego dokonać tylko w jeden sposób (tzn. wektory układu muszą być liniowo niezależne). Oznacza to, że dowolny zbiór n liniowo niezależnych wektorów przestrzeni n wymiarowej tworzy bazę tej przestrzeni. Mówimy wtedy, że wektory $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n$ rozpinają przestrzeń $\boldsymbol{V}$. Przestrzeń składająca się tylko z wektora zerowego $\boldsymbol{o} = [0, ..., 0]$ nie ma bazy, gdyż nie zawiera układu wektorów liniowo niezależnych.

Jeżeli przestrzeń liniowa $\boldsymbol{V}$ ma bazę skończoną, to liczba n wektorów bazy nazywa się jej wymiarem. Wymiar przestrzeni jest oznaczany jako $\dim \boldsymbol{V} = n$.

Można również wykazać, że dowolny zbiór wektorów liniowo niezależnych w przestrzeni liniowej można uzupełnić do bazy tej przestrzeni, oraz że każde dwie bazy przestrzeni $\boldsymbol{V}$ składają się zawsze z tej samej liczby wektorów [21, 31]. W zależności od układu wektorów przestrzeni mamy do czynienia z bazą ortogonalną lub ortonormalną tej przestrzeni.

Przykład 2.4. *Wektory* $\boldsymbol{a} = [1, 3, -2]$, $\boldsymbol{b} = [5, 1, 4]$ *oraz* $\boldsymbol{c} = [-1, 1, 1]$ *tworzą bazę ortogonalną liniowej przestrzeni euklidesowej* $\boldsymbol{E}^3$. *Jak wiadomo, bazę przestrzeni tworzą tylko wektory liniowo niezależne. Dla wykazania liniowej niezależności wektorów* $\boldsymbol{a}$, $\boldsymbol{b}$ *oraz* $\boldsymbol{c}$ *wystarczy stwierdzić, że równanie*:

$$\alpha_1 \boldsymbol{a} + \alpha_2 \boldsymbol{b} + \alpha_3 \boldsymbol{c} = \boldsymbol{o}$$

posiada jedynie rozwiązanie dla $\alpha_1 = \alpha_2 = \alpha_3 = 0$. *Można to sprawdzić, rozwiązując układ równań:*

$$\begin{aligned} \alpha_1 1 + \alpha_2 5 - \alpha_3 1 &= 0, \\ \alpha_1 3 + \alpha_2 1 + \alpha_3 1 &= 0, \\ -\alpha_1 2 + \alpha_2 4 + \alpha_3 1 &= 0. \end{aligned}$$

Powyższy układ równań można przedstawić w postaci macierzy:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 5 & -1 \\ 3 & 1 & 1 \\ -2 & 4 & 1 \end{bmatrix}.$$

Ponieważ wyznacznik $det(\boldsymbol{B}) \neq 0$, *zatem układ równań jest jednorodnym układem Cramera z jedynym rozwiązaniem* $\alpha_1 = \alpha_2 = \alpha_3 = 0$. *Wektory bazy* $\boldsymbol{a}$, $\boldsymbol{b}$ *oraz* $\boldsymbol{c}$ *są parami ortogonalne, gdyż np. iloczyn skalarny wektorów* $\langle \boldsymbol{a}, \boldsymbol{c} \rangle = 1 \cdot (-1) + 3 \cdot 1 - 2 \cdot 1 = 0$.
Podobne związki zachodzą dla iloczynów skalarnych wektorów $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \langle \boldsymbol{b}, \boldsymbol{c} \rangle = 0$. *Wektory nie są unormowane, gdyż* $||\boldsymbol{a}|| = \sqrt{14}$, $||\boldsymbol{b}|| = \sqrt{42}$ *oraz* $||\boldsymbol{c}|| = \sqrt{3}$.

Przykład 2.5. *Wektory* $\boldsymbol{a} = [-\frac{1}{2}, \frac{\sqrt{3}}{2}, 0]$, $\boldsymbol{b} = [\frac{\sqrt{3}}{2}, \frac{1}{2}, 0]$ *oraz* $\boldsymbol{c} = [0, 0, 1]$ *tworzą bazę ortonormalną liniowej przestrzeni euklidesowej* $\boldsymbol{E}^3$.

Liniową niezależność wektorów $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ *można sprawdzić identycznie jak w przykładzie poprzednim. Rozwiązanie wskazuje, że wektory* $\boldsymbol{a}$, $\boldsymbol{b}$ *oraz* $\boldsymbol{c}$ *są liniowo niezależne. Iloczyny skalarne* $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \langle \boldsymbol{a}, \boldsymbol{c} \rangle = \langle \boldsymbol{b}, \boldsymbol{c} \rangle = 0$, *tak więc odpowiednie wektory są parami ortogonalne. Wektory są też unormowane, ponieważ* $\|\boldsymbol{a}\| = \|\boldsymbol{b}\| = \|\boldsymbol{c}\| = 1$.

Spostrzeżenie 2.1. *Ortogonalność wektorów implikuje ich liniową niezależność. Niech* $\boldsymbol{V}$ *jest przestrzenią liniową. Wektory* $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n \in \boldsymbol{V}$, *z których każdy ma n elementów, tworzą bazę tej przestrzeni wtedy i tylko wtedy, gdy*

$$det \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_n \end{bmatrix}_{n \times n} \neq 0. \tag{2.24}$$

Wektory bazowe przestrzeni można więc zapisywać także w postaci macierzy.

Niech $\boldsymbol{A} = [a_{ij}]_{n \times n}$, gdzie $n \in \boldsymbol{N}$. Wówczas unormowane wektory wierszowe $\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n \in \boldsymbol{E}^n$ macierzy $\boldsymbol{A}$:

$$\begin{aligned} \boldsymbol{a}_1 &= [a_{11}, a_{12}, ..., a_{1n}], \\ \boldsymbol{a}_2 &= [a_{21}, a_{22}, ..., a_{2n}], \\ &\vdots \\ \boldsymbol{a}_n &= [a_{n1}, a_{n2}, ..., a_{nn}], \end{aligned} \tag{2.25}$$

tworzą bazę ortonormalną przestrzeni liniowej $\boldsymbol{E}^n$ wtedy i tylko wtedy, gdy macierz $\boldsymbol{A}$ jest macierzą nieosobliwą, $\boldsymbol{A}^{-1} = \boldsymbol{A}$, a iloczyn macierzy $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}_n$, $det(\boldsymbol{A}) = 1$. Bazę $\boldsymbol{E}^n$ tworzą również wektory kolumnowe macierzy $\boldsymbol{A}$. Można łatwo wykazać, że macierz $\boldsymbol{A}$ musi rzeczywiście posiadać takie własności. Wiersze macierzy $\boldsymbol{A}$ z założenia muszą być parami ortogonalne. Dotyczy to również kolumn tej macierzy. Z twierdzenia Cauchy'ego o wyznaczniku iloczynu macierzy [36] wynika, że $det(\boldsymbol{A}\boldsymbol{A}^T) = det(\boldsymbol{A}) \cdot det(\boldsymbol{A}^T)$.

Z własności wyznaczników wynika, że $det(\boldsymbol{A}) = det(\boldsymbol{A}^T)$, zatem $det(\boldsymbol{A}\boldsymbol{A}^T) = det(\boldsymbol{A}) \cdot det(\boldsymbol{A}) = [det(\boldsymbol{A})]^2$. Ponieważ macierz $\boldsymbol{A}$ jest ortogonalna, zatem $[det(\boldsymbol{A})]^2 = det(\boldsymbol{A}) = 1$. Macierz $\boldsymbol{A}$ jest więc macierzą nieosobliwą. Iloczyn macierzy $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}_n$ jest macierzą diagonalną (jednostkową). Macierz jednostkowa jest oczywiście nieosobliwa, gdyż $det(\boldsymbol{I}_n) = 1$. Jeśli wiersze macierzy $\boldsymbol{A}$ nie są unormowane, to macierz $\boldsymbol{A}$ tworzy bazę ortogonalną przestrzeni $\boldsymbol{E}^n$. Mamy wtedy $\boldsymbol{A}\boldsymbol{A}^T = \lambda \boldsymbol{I}_n$.

Przykład 2.6. *Dana jest macierz* $\boldsymbol{A}[a_{ij}]$:

$$\boldsymbol{A} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{2.26}$$

Łatwo również zauważyć, że $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$, $\boldsymbol{A} = \boldsymbol{A}^T$ *oraz* $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}_4$. *Macierz* $\boldsymbol{A}$ *jest więc symetryczna. Każda para wektorów wierszowych (kolumnowych) macierzy* $\boldsymbol{A}$ *jest ortogonalna, gdyż wybierając w sposób dowolny dwa wektory wierszowe (kolumnowe) macierzy* $\boldsymbol{A}$, *na przykład wektory* $\boldsymbol{a}_2 = [\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}]$ *oraz* $\boldsymbol{a}_4 = [\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}]$, *ich iloczyn skalarny wynosi* $\langle \boldsymbol{a}_2, \boldsymbol{a}_4 \rangle = \frac{1}{2} \cdot \frac{1}{2} + (-\frac{1}{2}) \cdot (-\frac{1}{2}) + \frac{1}{2} \cdot (-\frac{1}{2}) + (-\frac{1}{2}) \cdot \frac{1}{2} = 0$.
W identyczny sposób można postąpić dla każdej pary wektorów wierszowych (kolumnowych), otrzymując identyczne wyniki:

$$\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle = 0 \ \textit{dla} \ \ i, j = 1, ..., 4 \ \textit{oraz} \ i \neq j,$$

$$\langle \boldsymbol{a}_i, \boldsymbol{a}_i \rangle = ||\boldsymbol{a}_i|| = 1 \ \textit{dla} \ \ i = 1, ..., 4.$$

Zbiór wektorów wierszowych (kolumnowych) macierzy $\boldsymbol{A}$ *jest więc zbiorem wektorów ortogonalnych i unormowanych. Wektory* $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3, \boldsymbol{a}_4$ *są również liniowo niezależne, gdyż rozwiązanie równania:*

$$\alpha_1 \boldsymbol{a}_1 + \alpha_2 \boldsymbol{a}_2 + \alpha_3 \boldsymbol{a}_3 + \alpha_4 \boldsymbol{a}_4 = \boldsymbol{o},$$

które rozwinąć można do układu równań:

$$\begin{aligned}
\alpha_1\tfrac{1}{2}+\alpha_2\tfrac{1}{2}+\alpha_3\tfrac{1}{2}+\alpha_4\tfrac{1}{2} &= 0,\\
\alpha_1\tfrac{1}{2}-\alpha_2\tfrac{1}{2}+\alpha_3\tfrac{1}{2}-\alpha_4\tfrac{1}{2} &= 0,\\
\alpha_1\tfrac{1}{2}+\alpha_2\tfrac{1}{2}-\alpha_3\tfrac{1}{2}-\alpha_4\tfrac{1}{2} &= 0,\\
\alpha_1\tfrac{1}{2}-\alpha_2\tfrac{1}{2}-\alpha_3\tfrac{1}{2}+\alpha_4\tfrac{1}{2} &= 0,
\end{aligned}$$

jest możliwe tylko dla współczynników o wartościach $\alpha_1 = \alpha_2 = \alpha_3 = 0$. *Oznacza to, że wszystkie wektory wierszowe* (*kolumnowe*) *macierzy* $\boldsymbol{A}$ *tworzą bazę ortonormalną liniowej przestrzeni euklidesowej* $\boldsymbol{E}^4$.

Gdyby oznaczyć przez $\boldsymbol{B}$ macierz znajdującą się po prawej stronie równania (2.26), w której występują tylko wartości ± 1, to zbiór wektorów wierszowych tej macierzy jest także zbiorem wektorów ortogonalnych, gdyż $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = 0$ dla $i, j = 1, ..., 4$ oraz $i \neq j$, ale $\langle \boldsymbol{b}_i, \boldsymbol{b}_i \rangle = 4$ oraz $||\boldsymbol{b}_i|| = 2$, dla $i = 1, ..., 4$. Wtedy $det(\boldsymbol{B}) = 16$, $\boldsymbol{B} = \boldsymbol{B}^T$, $\boldsymbol{B}^{-1} = \frac{1}{4}\boldsymbol{B}^T$, a iloczyn macierzy $\boldsymbol{B}\boldsymbol{B}^T = \frac{1}{4}\boldsymbol{I}_n$. Z punktu widzenia wymagań dotyczących ortogonalności macierzy macierz $\boldsymbol{B}$ nie jest ortogonalna, gdyż nie zachodzą dla niej związki $\boldsymbol{B}^{-1} = \boldsymbol{B}^T$ oraz $\det(\boldsymbol{B}) = 1$. Wektory $\boldsymbol{b}_i$ są wektorami nieunormowanymi, ale wszystkie mają tę samą długość (różną od 1). Wektory macierzy $\boldsymbol{B}$ rozpinają liniową przestrzeń euklidesową $\boldsymbol{E}^4$.

3. Preliminaria

Opisywane w tym rozdziale zagadnienia są znane i zostały omówione w literaturze, nie wnoszą więc nowych treści. Przyjęcie takiej formuły ich prezentacji pozwala Czytelnikowi skupić się na analizie opisywanych w kolejnych rozdziałach zagadnień, bez konieczności poszukiwania i studiowania literatury dodatkowej. Niezbędne pozycje literaturowe są oczywiście w odpowiednich miejscach przywoływane, pozwalając w razie potrzeby na lekturę dodatkowych prac.

3.1. Arytmetyka modularna. Kongruencje

Arytmetyka modularna, nazywana także arytmetyką reszt, jest podobna do zwykłej arytmetyki na liczbach całkowitych z tą różnicą, że obliczenia prowadzone są modulo $n \pmod{n}$.

Definicja 3.1. *Niech n będzie liczbą naturalną ($n \in \boldsymbol{N}$) oraz niech a oraz b będą liczbami całkowitymi ($a, b, k \in \boldsymbol{Z}$). Mówimy, że a przystaje do b modulo n, jeżeli różnica $a - b$ jest podzielna przez n, co symbolizuje zapis $a \equiv b \pmod{n}$ lub $a \equiv_n b$:*

$$a \equiv b \pmod{n} \Leftrightarrow \bigvee_{k \in \boldsymbol{Z}} a - b = k \cdot n. \tag{3.1}$$

Liczbę n nazywamy modułem kongruencji.

Zgodnie z Definicją (3.1), kongruencja to sposób zapisu informującego, że dwie liczby całkowite a oraz b dają tę samą resztę przy dzieleniu każdej z nich przez pewną liczbę naturalną n. Inaczej: liczby a i b przystają modulo n (są kongruentne), jeśli ich różnica $a - b$ dzieli się bez reszty przez n.

Notacja $a \equiv b \pmod{n}$ oznacza, że dla pewnej liczby $k \in \boldsymbol{Z}$ zachodzi związek $a = b + k \cdot n$. Zapis $a \pmod{n}$ odnosi się do operacji tzw. redukcji modularnej, która oznacza, że reszta z dzielenia a/n spełnia warunek $a/n < n$.

Formalny opis arytmetyki modularnej przedstawić można za pomocą teorii grup. Grupę $(A, \oplus_n)$ tworzy zbiór A wraz ze definiowaną w A operacją $\oplus_n$ posiadającą następujące własności:

– Dla $a, b \in A$ element $a \oplus_n b \in A$.
– Istnieje element $e \in A$ zwany elementem neutralnym taki, że $e \oplus_n a = a \oplus_n e = a$ dla dowolnego $a \in A$.
– Dla $a, b, c \in A$ zachodzi $(a \oplus_n b) \oplus_n c = a \oplus_n (b \oplus_n c)$.
– Dla elementu $a \in A$ istnieje dokładnie jeden element odwrotny $b \in A$ taki, że $a \oplus_n b = b \oplus_n a = e$.

Zgodnie z powyższym, można wprowadzić w zbiorze $A = \{0, 1, ..., n-1\}$ arytmetykę modulo n, definiując działania:

– dodawania:

$$a + b = \begin{cases} a + b, & \text{dla } a + b < n \\ a + b - n, & \text{dla } a + b \geq n \end{cases}, \tag{3.2}$$

– odejmowania:

$$a - b = \begin{cases} a - b, & \text{dla } a - b \geq 0 \\ a - b + n, & \text{dla } a - b < 0 \end{cases}, \tag{3.3}$$

– wyznaczenia liczby przeciwnej do liczby a:

$$c = \begin{cases} -a + n, & \text{dla } a > 0 \\ 0, & \text{dla } a = 0 \end{cases}, \tag{3.4}$$

– wyznaczenia liczby odwrotnej do liczby $a \pmod n$. Jest to taka liczba x, że:

$$ax \equiv 1 \pmod n. \tag{3.5}$$

Liczbę x zapisuje się często w postaci:

$$x = [a^{-1}]_n. \tag{3.6}$$

Program 3.1. *Liczbę odwrotną modulo n do liczby a można wyznaczyć za pomocą programu Matlab.*

```
% Obliczanie odwrotnosci a modulo n
a=13;                % Liczba a
n=8;                 % Liczba modulo
p0=0; p1=1;          % Zmienne pomocnicze
a0=a; n0=n;          % Zmienne pomocnicze
q=floor(n0/a0);      % Zwraca najwieksza liczbe calkowita,
                     % mniejsza lub rowna n0/a0
r=mod(n0,a0);        % Reszta z dzielenia n0/a0
while (r>0)          % Glowna petla programu. Warunek zakonczenia
      t=p0-q*p1;
      if (t>=0)
         t=mod(t,n);
      else
         t=n-(mod(-t,n));
      end
      p0=p1; p1=t;
      n0=a0; a0=r;
      q=floor(n0/a0);
      r=mod(n0,a0);
end
p1                   % Wyznaczona liczba
```

3.2. Zwykły i rozszerzony algorytm Euklidesa

Za pomocą zwykłego algorytmu Euklidesa wyznacza się największy wspólny dzielnik (NWD) dwóch liczb naturalnych a oraz b. Jest to powszechnie znany algorytm, implementowany w różnych językach programowania. Jego opis jest więc zbędny. Algorytm Euklidesa jest implementowany w wielu pakietach matematycznych.

Znalezienie największego wspólnego dzielnika (NWD) liczb a oraz b umożliwia na przykład polecenie Matlaba **gcd**(a,b), a polecenie **mod**(a,b) wyznacza resztę z dzielenia a/b.

W rozszerzonym algorytmie Euklidesa, oprócz znajdowania $NWD(a,b)$, $a,b>0$, wyznaczane są również takie liczby całkowite s,t, że [8,29]:

$$NWD(a,b) = a \cdot s + b \cdot t. \tag{3.7}$$

Istnienie takich liczb udowadnia lemat E. Bezout, który rozstrzyga, że tożsamość (3.7) jest zawsze prawdziwa. W praktycznej realizacji algorytmu Euklidesa wykorzystuje się spostrzeżenie, że dla $a>b$ i $a,b \in \boldsymbol{N}$ zachodzi związek:

$$NWD(a,b) = NWD((a \bmod b), b), \tag{3.8}$$

gdzie $a \bmod b$ oznacza resztę z dzielenia a/b.

Program 3.2. *Iteracyjna realizacja rozszerzonego algorytmu Euklidesa, gdzie wykorzystuje się zależność* (3.8). *Dla danych wejściowych* (*liczby* A,B) *algorytm wyznacza liczby* s, t, *zgodnie z formułą* (3.7).

```
a=A; b=B;                          % A i B dane wejsciowe
s=1; t=0; r=0; u=1;                % Zmienne pomocnicze
while a<0 | a~=round(a)            % Kontrola poprawnosci liczby a
      a=input('podaj wartosc a:');
end;
while b<0 | b~=round(b)            % Kontrola poprawnosci liczby b
      b=input('podaj wartosc b:');
end;
while (b~=0)                     % Glowna petla programu
      c=mod(a,b);                  % Reszta z dzielenia a/b
      d=floor(a/b);                % Czesc calkowita z dzielenia a/b
      a=b;                         % Zapamietanie biezacej wartosci b
      b=c;                         % Zapamietanie biezacej wartosci c
```

```
        n_r=s-d*r;
        n_u=t-d*u;
        s=r;
        t=u;;
        r=n_r;
        u=n_u;
end
s
t
```

Algorytm Euklidesa można zrealizować w dwóch technikach programowania – w wersji iteracyjnej lub rekurencyjnej.

3.3. Chińskie twierdzenie o resztach

Chińskie twierdzenie o resztach (ang. *Chinese remainder theorem* – CRT) mówi, że określony układ kongruencji spełnia dokładnie jedna liczba.

Twierdzenie to jest często wykorzystywane w algorytmach kryptograficznych, gdyż pozwala na wykonywanie operacji matematycznych na dużych liczbach.

Twierdzenie 3.1. (*chińskie twierdzenie o resztach*). *Niech* $M = \Pi_{i=1}^{k} m_i$ *jest iloczynem liczb względnie pierwszych,* $m_1, ..., m_k \in \boldsymbol{N} \setminus \{0\}$*, co oznacza, że* $NWD(m_i, m_j) = 1$*, dla* $i \neq j$*.*
Niech $a_1, ..., a_k \in \boldsymbol{N}$ *będą dowolnymi liczbami. Wtedy układ równań diofantycznych jest układem kongruencji*:

$$c \equiv a_i \pmod{m_i}, \quad i = 1, ..., k, \tag{3.9}$$

i ma dokładnie jedno rozwiązanie c *takie, że*:

$$c = \sum_{i=1}^{k} \frac{M}{m_i} \cdot s_i \cdot a_i, \qquad \pmod{M}, \tag{3.10}$$

gdzie liczba s_i *jest rozwiązaniem kongruencji typu* $\frac{M}{m_i} \cdot c \equiv 1 \pmod{m_i}$, *dla* $i = 1, ..., k$.

Przykład 3.1. *Korzystając z chińskiego twierdzenia o resztach, należy wyznaczyć dodatnie rozwiązanie układu kongruencji:*

$$\begin{cases} c \equiv 4 & (\text{mod } 3) \\ c \equiv 7 & (\text{mod } 10) \\ c \equiv 6 & (\text{mod } 7) \end{cases}.$$

Ponieważ $NWD(3, 10) = NWD(3, 7) = NWD(10, 7) = 1$, *zatem liczby* $3, 10$ *oraz* 7 *są względnie pierwsze. Układ kongruencji ma więc rozwiązanie. Otrzymujemy* $M = 3 \cdot 10 \cdot 7 = 210$ *oraz:*

$$\begin{aligned} \frac{M}{m_1} &= \frac{210}{3} = 70, \\ \frac{M}{m_2} &= \frac{210}{10} = 21, \\ \frac{M}{m_3} &= \frac{210}{7} = 30. \end{aligned}$$

Z podstaw arytmetyki modularnej wiadomo, że zachodzi związek: $NWD(a, b) = a \cdot s + b \cdot t$, *gdzie* s *i* t *są pewnymi liczbami całkowitymi. Jeśli znane są liczby* a *oraz* b, *tutaj znane są pary liczb:* $(70, 3)$ $(21, 10)$ *oraz* $(30, 7)$, *to zestawy liczb* s *i* t *można wyznaczyć za pomocą rozszerzonego algorytmu Euklidesa, który był już opisany wcześniej*:

$70 \cdot c \equiv 1 \pmod{3} : NWD(70, 3) = 1 = 70 \cdot 1 + 3 \cdot (-23) \rightarrow s_1 = 1,$
$21 \cdot c \equiv 1 \pmod{10} : NWD(21, 10) = 1 = 21 \cdot 1 + 10 \cdot (-2) \rightarrow s_2 = 1,$
$30 \cdot c \equiv 1 \pmod{7} : NWD(30, 7) = 1 = 30 \cdot (-3) + 7 \cdot (13) \rightarrow s_3 = -3.$

Wtedy, na podstawie chińskiego twierdzenia o resztach, otrzymujemy:
$c = 70 \cdot 1 \cdot 4 + 21 \cdot 1 \cdot 7 + 30 \cdot (-3) \cdot 6 = -113 \pmod{210}$
lub w innym zapisie: $-113 \equiv_{210} 97$, *bo* $-113 \pmod{210} = 97$.

Rozwiązaniem zbioru kongruencji jest więc liczba $c = 97$.
W Matlabie można łatwo sprawdzić, że warunki zadanego układu kongruencji są spełnione, gdyż: $\mathbf{mod}(97, 3) = \mathbf{mod}(4, 3) = 1, \mathbf{mod}(97, 10) = \mathbf{mod}(7, 10) = 7$ *oraz* $\mathbf{mod}(97, 7) = \mathbf{mod}(6, 7) = 6$.

3.4. Notacja O

Wiele współczesnych naukowych i technologicznych zagadnień jest tak skomplikowanych, że wymagają stosowania komputerów. Jednocześnie zmierza się do tego, aby komputery działały szybko, czyli by szybkość obliczeń była duża. Aby zwiększyć szybkość działania procesorów, niezbędne jest udoskonalanie technologii projektowania i wytwarzania układów mikroelektronicznych o dużym stopniu scalenia. Produkowane obecnie układy o ultrawysokim stopniu scalenia (ang. *Ultra Large Scale of Integration*) są tak wydajne, że szerokość ścieżek łączących wewnętrzne elementy układu jest bliska wymiarom atomowym, co może stanowić barierę dalszego postępu technologicznego. Pewnym ograniczeniom podlegają również próby podnoszenia częstotliwości taktowania układów, gdyż pojawiają się wtedy kłopotliwe do wyeliminowania opóźnienia czasowe zakłócające współpracę elementów wewnętrznych. Innym rodzajem eksperymentów zmierzających do zwiększenia szybkości działania układów są próby różnicowania napięć zasilających układ. Wyrafinowana technologia nie zwalnia jednak od poszukiwań takich rozwiązań programowych, które z zastosowaniem danego sprzętu będą wykonywane najszybciej. Badaniem algorytmów zajmuje się algorytmika.

Niektóre algorytmy są szybsze od innych. Wszystkie jednak wymagają więcej czasu, by operować na większej liczbie danych wejściowych. Jeśli liczba danych wejściowych nie będzie duża, to prawdopodobnie nie zauważy się różnicy w czasie działania różnych algorytmów na danej maszynie, ale dla zadania operującego na dużej liczbie danych wybór właściwego algorytmu ma znaczenie kluczowe – może prowadzić do istotnego skrócenia czasu obliczeń. Sposobem oceny algorytmu może być szacowanie jego złożoności czasowej. Złożoność czasowa jest elementem złożoności obliczeniowej, w przypadku której badany jest

czas wykonania lub zapotrzebowanie algorytmu na pamięć operacyjną. Miarą złożoności czasowej jest liczba podstawowych (dominujących) operacji niezbędnych do zrealizowania algorytmu. Pomiar rzeczywistego czasu byłby mylący, ze względu na silną zależność realizacji algorytmu od użytego kompilatora oraz sprzętu komputerowego, którym dysponujemy. Znając czas wykonania elementarnej operacji na danym komputerze, złożoność czasową można określać również w jednostkach czasu, czego jednak się nie praktykuje.

Projektując algorytm, chcemy, żeby jego złożoność obliczeniowa była jak najmniejsza, czyli aby program liczył się jak najkrócej i wykorzystywał jak najmniej zasobów pamięci operacyjnej. Dążymy więc do minimalizowania kosztów algorytmu. Nie zawsze takie rozumowanie jest jednak uprawnione – często lepiej jest dobrać prostszy algorytm niż poszukiwać algorytmu szybkiego, lecz bardziej złożonego, a tym samym trudniejszego do zapisania w wybranym języku programowania, co może być przyczyną powstawania trudnych do znalezienia i eliminacji błędów wykonania.

Definicja 3.2. *Niech t i g będą ciągami liczb rzeczywistych. Zapisujemy, że*:

$$t(n) = O(g(n)) \tag{3.11}$$

wtedy, gdy istnieją stałe $c, n_0 \in \boldsymbol{R}^+$ takie, że:

$$\bigwedge_{n>n_0} t(n) \leq c \cdot g(n), \tag{3.12}$$

gdzie n jest rozmiarem danych wejściowych.
Mówimy wtedy, że funkcja t jest co najwyżej rzędu g.

Równość $t(n) = O(g(n))$ należy interpretować w ten sposób, że funkcja $t(n)$ należy do zbioru wszystkich funkcji, które spełniają tę równość. Definicja 3.2 wskazuje, że dla dużych n wartości t nie są większe niż wartości g pomnożone przez pewną stałą, czyli funkcja $t(n)$ jest dla wszystkich n większych od n_0 ograniczona przez funkcję $g(n)$ pomnożoną przez pewną stałą c. Ten sposób szacowania złożoności powoduje eliminowanie składników wolniej

rosnących oraz zaniedbywanie stałych, przez które mnożone są funkcje. Nie zawsze jest to korzystne. Załóżmy, że mamy dwa algorytmy, które rozwiązują to samo zadanie.

Niech pierwszy algorytm ma złożoność $O(10^{12}n)$, a drugi $O(n^2)$. Liczba operacji elementarnych, szacowanych notacją O, w algorytme drugim wydaje się większa niż w algorytmie pierwszym. Tak jest rzeczywiście dopiero dla odpowienio dużych n: $n > 10^{12}$.

Może się jednak zdarzyć, że algorytm wykonuje różną liczbę iteracji w zależności od konkretnego zestawu danych wejściowych. Wtedy możemy badać czas jego wykonania w najgorszym przypadku, czyli gdy przetwarzamy najgorszy z możliwych zestawów danych. Mówimy wtedy o złożoności pesymistycznej.

Spośród wielu różnych złożoności czasowych algorytmów najbardziej pożądane są te, których złożoność jest rzędu $O(1)$, $O(n)$ lub $O(n \log_2 n)$, a więc o złożoności stałej, liniowej lub liniowo-logarytmicznej. Najwolniej wykonywanymi algorytmami są algorytmy o złożoności wielomianowej $O(n^x)$, gdzie x jest dodatnią stałą całkowitą, i algorytmy o złożoności wykładniczej $O(x^n)$ dla $x > 2$. Prezentowane w kolejnych rozdziałach książki algorytmy charakteryzowane są między innymi za pomocą notacji O.

4. Dyskretne reprezentacje deterministycznych sygnałów ciągłych

W świecie rzeczywistym mierzone sygnały mają charakter ciągły – są to więc sygnały analogowe. W opisie sygnałów analogowych wykorzystuje się przestrzenie funkcyjne. Reprezentacja sygnałów analogowych w kategoriach przestrzeni funkcyjnych jest dobrze ugruntowana teoretycznie, pozwalając na rozwiązywanie praktycznych zagadnień teorii sygnałów. Pod wieloma względami przestrzenie funkcyjne są podobne do przestrzeni wektorowych, co pozwala przenieść obliczenia do tych przestrzeni. Podstawowe pojęcia przestrzeni funkcyjnej mają jednak ograniczenia, których nie ma w przypadku przestrzeni wektorowych. Podstawą rozpatrywanych w tym rozdziale przestrzeni funkcyjnych jest założenie, że funkcje muszą być całkowalne z kwadratem. Warunek ten wynika z faktu, że podobnie jak w przestrzeniach wektorowych, także w przestrzeniach funkcyjnych obowiązuje pojęcie iloczynu skalarnego i normy.

Definicja 4.1. *Iloczynem skalarnym rzeczywistych, ciągłych funkcji $f(x)$ i $g(x)$ zmiennej rzeczywistej, określonych na przedziale I, nazywamy całkę*:

$$\langle f(x), g(x) \rangle = \int_I f(x) \cdot g(x) dx. \tag{4.1}$$

Suma iloczynów współrzędnych wektorów (Definicja 2.3) została więc zastąpiona całką iloczynu funkcji.

Definicja 4.2. *Normą funkcji $f(x)$, określonej na przedziale I, nazywamy liczbę*:

$$\|f(x)\| = \sqrt{\langle f(x), f(x) \rangle} = \sqrt{\int_I f^2(x) dx}. \tag{4.2}$$

Aby można było wyznaczyć normę funkcji, musi istnieć całka z jej kwadratu, co wynika bezpośrednio ze wzoru (4.2). Jeżeli nie jest to doprecyzowane, to najczęściej ma się na uwadze funkcje całkowalne w sensie Lebesgue'a. Trzeba pamiętać, że nie każda funkcja spełnia ten ważny postulat.

Przykład 4.1. *Można policzyć całkę* $\int\limits_0^1 \frac{1}{\sqrt{x}}dx = 2\sqrt{x}|_0^1 = 2$, *co wynika z faktu, że* $\int x^a dx = \frac{1}{a+1}x^{a+1} + C$, $a \neq -1$.
Natomiast dla funkcji $\left(\frac{1}{\sqrt{x}}\right)^2$ *jej całka* $\int\limits_0^1 \left(\frac{1}{\sqrt{x}}\right)^2 dx = \int\limits_0^1 \frac{1}{x}\,dx = +\infty$, *gdyż całka* $\int\limits_0^1 \frac{1}{x}\,dx = \lim\limits_{\varepsilon\to 0^+}\int\limits_\varepsilon^1 \frac{1}{x}dx = \lim\limits_{\varepsilon\to 0^+}(\ln 1 - \ln\varepsilon) = \lim\limits_{\varepsilon\to 0^+}(-\ln\varepsilon) = +\infty$.

Definicja 4.3. *Funkcje* $f(x)$ *oraz* $g(x)$ *są ortogonalne na przedziele* I, *jeśli*:

$$\langle f(x), g(x)\rangle = \int\limits_I f(x)\cdot g(x)dx = 0. \tag{4.3}$$

Przykład 4.2. *Funkcjami ortogonalnymi w przedziale* $[-\pi,\pi]$ *są funkcje* $\sin(nx)$ *oraz* $\cos(mx)$ *dla* n,$m \in \boldsymbol{C}$ *i* $n \neq m$. *Funkcje te są całkowalne z kwadratem*:

$$\|\sin(nx)\| = \sqrt{\int\limits_{-\pi}^{\pi}\sin^2(nx)dx} = \sqrt{\pi},$$

$$\|\cos(mx)\| = \sqrt{\int\limits_{-\pi}^{\pi}\cos^2(mx)dx} = \sqrt{\pi}.$$

Z trygonometrii wiadomo, że:

$$\sin(\alpha)+\sin(\beta) = 2\sin\tfrac{1}{2}(\alpha+\beta)\cos\tfrac{1}{2}(\alpha-\beta).$$

Niech $\frac{1}{2}(\alpha+\beta) = nx$ *i* $\frac{1}{2}(\alpha-\beta) = mx$. *Jest to układ równań, w którym poszukiwanymi wartościami są* α *oraz* β. *Rozwiązując ten układ, otrzymujemy* $\alpha = (n+m)x$ *i* $\beta = (n-m)x$, *stąd*:

$$\sin(n+m)x + \sin(n-m)x = 2\sin(nx)\cdot\cos(mx),$$
$$\tfrac{1}{2}\sin(n+m)x + \sin(n-m)x = \sin(nx)\cdot\cos(mx).$$

Ostatecznie ortogonalność funkcji $\sin(nx)$ *oraz* $\cos(mx)$ *wynika z faktu, że*:

$$\langle \sin(nx), \cos(mx) \rangle = \int_{-\pi}^{\pi} \sin(nx)\cos(mx)dx =$$

$$= \frac{1}{2}\int_{-\pi}^{\pi} (\sin(n+m)x + \sin(n-m)x)dx =$$

$$= -\frac{1}{2}\frac{\cos(n+m)x}{n+m} - \frac{1}{2}\frac{\cos(n-m)x}{n-m}|_{-\pi}^{\pi} = 0.$$

Przykład 4.3. *Układem ortogonalnym w przedziale* $[0, \pi]$ *jest również układ funkcji* $\sin(x)$, $\sin(2x)$,..., $\sin(nx)$, *dla* $n \in \boldsymbol{N}$. *Wystarczy zauważyć, że*:

$$\|\sin(nx)\| = \sqrt{\int_0^{\pi} \sin^2(nx)dx} = \sqrt{\frac{\pi}{2}}.$$

Wiadomo, że $\sin(\alpha)\sin(\beta) = \frac{1}{2}cos(\alpha - \beta) - \frac{1}{2}cos(\alpha + \beta)$. *Podstawiając* $\alpha = nx$ *oraz* $\beta = mx$, *otrzymujemy*:

$$\sin(nx)\sin(mx)\rangle = \frac{1}{2}\int_0^{\pi} (\cos(n-m)x - \cos(n+m)x)dx =$$

$$= \frac{1}{2}\left(\frac{sin(n-m)x}{n-m} - \frac{sin(n+m)x}{n+m}\right)|_0^{\pi} = 0.$$

Na podstawie przeprowadzonych analiz można wykazać, że układem ortogonalnym w przedziale $[-\pi, \pi]$ jest układ następujących funkcji trygonometrycznych: $1, \cos x, \sin x, \cos 2x, \sin 2x, ..., \cos nx, \sin nx$. Układami ortogonalnymi są również układy wielomianów Legendre'a, Czybyszewa, układ funkcji Haara, Rademachera i inne [14, 34, 40]. Znanych jest bardzo wiele układów funkcji ortogonalnych, które mają duże znaczenie w teorii sygnałów [3, 28, 40] – nie o tym jednak traktuje ta monografia.

Przedmiotem zainteresowania większości aplikacji inżynierskich są nie tylko rzeczywiste sygnały ciągłe (lub sygnały ciągłe modelowane różnymi funkcjami), ale również sygnały dyskretyzowane w czasie, czyli ciągi liczb o skończonej długości. Tym sposobem funkcja zmiennej ciągłej jest reprezentowana odpowiednim ciągiem liczb. Ciąg ten stanowi reprezentację dyskretną funkcji ciągłej. Może być również ciągiem liczb określonych w jakiś inny sposób. Na przykład ciągiem takim może być ciąg liczb całkowitych lub rzeczywistych, bez konieczności wiązania tego ciągu z jakimkolwiek sygnałem ciągłym. Należy również pamiętać, że danemu ciągowi próbek może odpowiadać nieskończenie wiele sygnałów ciągłych.

Załóżmy, że mierzona jest pewna wielkość fizyczna w czasie, ze stałym odstępem czasu równym 2^{-m}. Oznacza to, że w jednostce czasu dokonujemy 2^m pomiarów. Zmierzone wartości tworzą ciąg $(...x_0, x_1, ..., x_{2^m-1}...)$. Można założyć, że ciąg ten jest z obu stron nieskończony, przyjmując, że przed i po zakończeniu pomiarów zmierzone wartości były stale równe 0. Tak rozpatrywany ciąg stanowi reprezentację dyskretną rozpatrywanej wielkości fizycznej, gdyż zawiera skończoną liczbę elementów różnych od zera. Zakładamy przy tym, że sygnał nie zmienia wartości pomiędzy pomiarami. Oznacza to, że funkcja jest stała w przedziałach $[n2^{-m}, (n+1)2^{-m}]$, $n = 0, 1, ..., 2^m - 1$. Przestrzenie sygnałów mogą być więc traktowane jako $N = 2^m$ wymiarowe przestrzenie wektorowe, w których uporządkowane kolejno w czasie próbki sygnału ciągłego są reprezentowane przez kolejne współrzędne wektora, oznaczone jako $[x(0), x(1), ...] = [x_0, x_1, ...]$.

Konsekwencją dyskretyzacji w pewnym przedziale I sygnału ciągłego $x(t)$ jest zastąpienie czasu jego indeksem oraz całki sumą:

$$x(t) \to x_t,\ t = 0, ..., N-1, \tag{4.4}$$

$$\int_I x(t)dt \to \sum_{t=0}^{N-1} x_t. \tag{4.5}$$

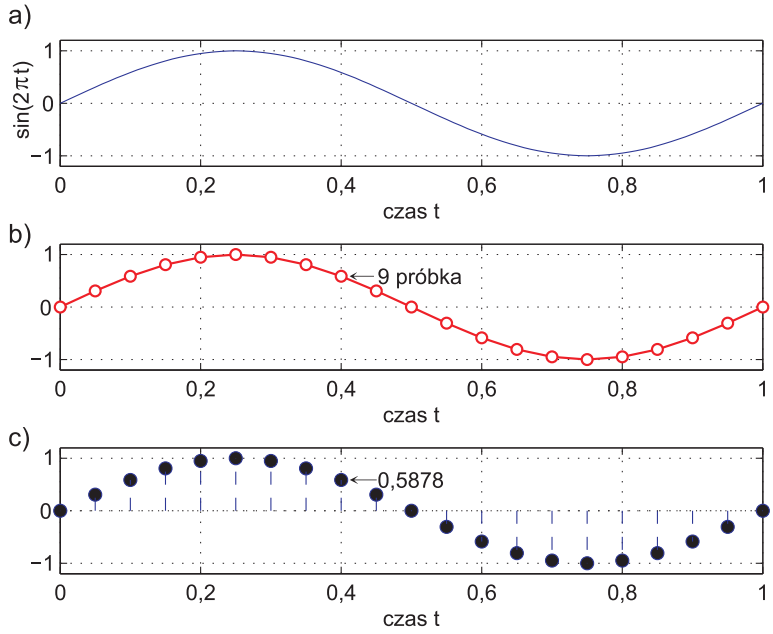
W ten sposób rozważania o funkcjach ciągłych mogą być rozszerzone na przestrzenie sygnałów dyskretnych, czyli przestrzenie wektorowe.

Na rys. 1 przedstawiono wykres prostej funkcji $y = \sin(2\pi t)$, a na kolejnych wykresach jej dyskretną reprezentację. Oznaczono także miejsca pobierania próbek tego sygnału oraz wartości poszczególnych próbek – dziewiąta próbka ma np. wartość $\sin(2\pi * 0.4) = 0.5878$. Sygnał był próbkowany z okresem $T = 0.05$. Poszczególne wykresy sprządzone zostały za pomocą programu Matlab.

Program 4.1. *Wykres funkcji trygonometrycznej* $y = \sin(2\pi t)$ *oraz jej kolejne reprezentacje.*

```
T=1; N=100; dt=T/N;
t=0:dt:(N)*dt; y = sin(2*pi*t);
subplot(311); plot(t,y); grid;
xlabel('czas t'); axis([0 1 -1.2 1.2]);
ylabel('sin(2\pit)');
t1=0:5*dt:(N)*dt;     y = sin(2*pi*t1);
subplot(312); plot(t1,y,'-ro','LineWidth',1,...
'MarkerEdgeColor','r',...
'MarkerFaceColor','w',...
'MarkerSize',5); grid
text(0.41,sin(0.8*pi),'\leftarrow 9 probka',...
'HorizontalAlignment','left');
xlabel('czas t'); axis([0 1 -1.2 1.2]);
subplot(313);
t1=0:5*dt:(N)*dt;
h = stem(t1,sin(2*pi*t1),'fill','--');
set(get(h,'BaseLine'),'LineStyle',':')
set(h,'MarkerFaceColor','black');grid
text(0.41,sin(0.8*pi),'\leftarrow 0.5878',...
'HorizontalAlignment','left');
xlabel('czas t'); axis([0 1 -1.2 1.2]);
```

Często zamiast korzystać z bezpośredniej reprezentacji funkcyjnej korzysta się z pewnej reprezentacji sygnału.



Rys. 1. Wykres funkcji (a): miejsca pobierania próbek (b) oraz wartości próbek w poszczególnych miejscach (c)

W ten sposób otrzymujemy dyskretne wartości analogowego sygnału pierwotnego (ciągłego). Wartości dyskretne można zapisać w postaci wektora.

Rozmiar takiego wektora jest równy liczbie pobranych próbek. Próbkowanie jest więc procesem pobierania w ustalonych odstępach czasu T próbek sygnału.

Parametr T jest okresem próbkowania, a parametr $1/T$ częstotliwością próbkowania. Otrzymujemy w ten sposób sygnał spróbkowany $f^*(n)$ sygnału ciągłego $f(t)$ taki, że $f^*(n) = f(nT)$. Oznacza to, że sygnał $f^*(n)$ jest ciągiem liczb będących wartościami funkcji $f(t)$ w punktach $t = nT$. Jak widać, dyskretyzacja sygnału ciągłego wiąże się z utratą części informacji o tym sygnale. Możliwość najdokładniejszego odtworzenia sygnału ciągłego na podstawie jego dyskretnej reprezentacji zapewnia fundamentalne twierdzenie o próbkowaniu – twierdzenie Kotielnikowa–Shannona [40]. Zgodnie z nim, jeśli sygnał ciągły nie posiada składowych widma o częstotliwości równej lub większej niż f,

to może zostać odtworzony z ciągu próbek tworzących sygnał dyskretny, jeśli próbki te zostały pobrane w odstępach czasowych nie większych niż $1/2f$, co oznacza, że częstotliwość f_g próbkowania sygnału ciągłego powinna wynosić co najmniej $f_g \geq 2f$. Wszystkie harmoniczne o częstotliwościach przekraczających wartość $f_g/2$ muszą być z sygnału odfiltrowane filtrem dolnoprzepustowym. Postulaty tego twierdzenia znane są również jako twierdzenie Whittakera–Nyquista i stanowią ważny element teorii cyfrowego przetwarzania sygnałów. Oba twierdzenia są podobne. Twierdzenie Kotielnikowa–Shannona rozstrzyga, kiedy z danego sygnału dyskretnego można odtworzyć sygnał ciągły. Twierdzenie Whittakera–Nyquista mówi o tym, jaka powinna być częstotliwość próbkowania, gdy znamy najwyższą składową występującą w sygnale pierwotnym. Twierdzenie Kotielnikowa–Shannona jest twierdzeniem bardzo mocnym. Pozwala odtworzyć wartości funkcji pasmowo ograniczonej w nieprzeliczalnie wielu punktach na podstawie znajomości wartości jej dyskretnych punktów. Zasady próbkowania sygnału ciągłego są podstawą działania każdego współczesnego urządzenia cyfrowego. Jeśli funkcja (sygnał pierwotny) s zawiera tylko częstotliwości mniejsze od f_g, a próbki wartości funkcji s są pobierane w chwilach oddalonych o $1/2f_g$, to funkcja s może być określona w dowolnej chwili za pomocą następującego wzoru interpolacyjnego:

$$s(t) = \sum_{n=-\infty}^{+\infty} s\left(\frac{n}{2f_g}\right) \frac{\sin(\pi(2f_g t - n)}{\pi(2f_g t - n)}. \tag{4.6}$$

Wzór (4.6) można przekształcić do postaci równoważnej. Wprowadzając oznaczenie $\Delta t = 1/2f_g$, otrzymujemy formułę:

$$s(t) = \sum_{n=-\infty}^{+\infty} s\left(n\Delta t\right) \frac{\sin(\pi(t - n\Delta t)/\Delta t)}{\pi(t - n\Delta t)/\Delta t}, \tag{4.7}$$

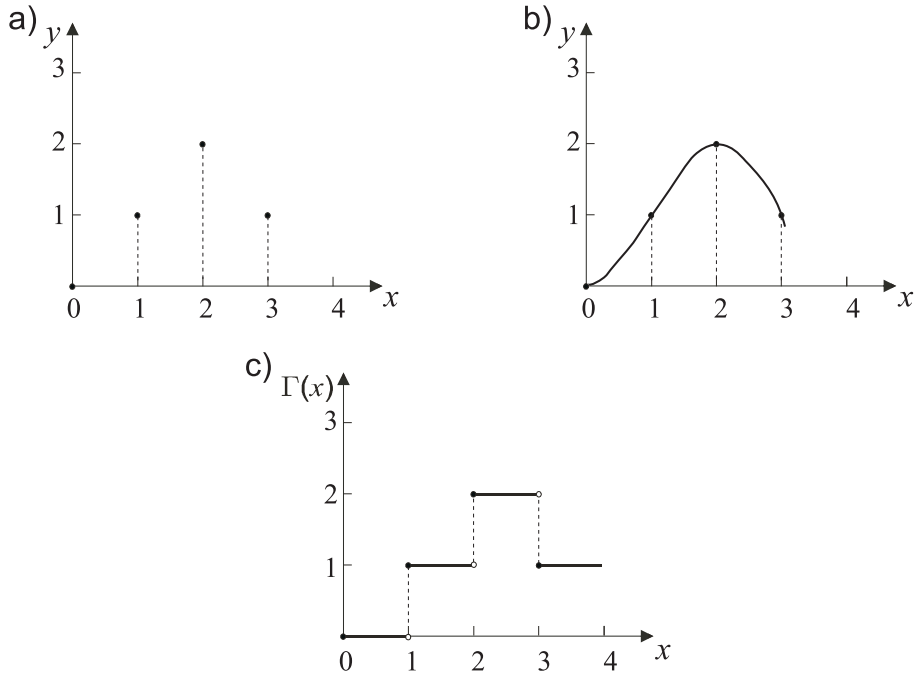
za pomocą której można aproksymować funkcję $s(t)$.

Zagadnienia odtwarzania sygnału oraz zjawiska powstające przy podpróbkowaniu lub nadpróbkowaniu nie są jednak przedmiotem naszego zainteresowania.

Dokładną analizę zjawisk występujących podczas próbkowania sygnału analogowego znaleźć można między innymi w pracach [34, 40].

Funkcje dyskretne mogą mieć różne reprezentacje w przestrzeniach funkcyjnych. Aby wartości funkcji dyskretnych były jednoznacznie odwzorowywane, ich reprezentacje w przestrzeni funkcyjnej muszą być odpowiednio interpolowane. Jeżeli przebieg funkcji $y_i = f(x_i)$ znany jest tylko dla określonej liczby argumentów x_i, $i = 1, ..., n$, to zadanie interpolacji sprowadza się do znalezienia funkcji interpolującej F takiej, że $y_i = F(x_i)$. Punkty x_1,..., x_n nazywane są węzłami interpolacji. Są to więc dane dyskretne. Dla realizacji tego zadania stosowane są różne techniki numeryczne, na przykład wielomiany, interpolacja kawałkami liniowa, kawałkami sześcienna lub funkcje odcinkowo-stałe [4,9,14,34]. W ogólności przez funkcję sklejaną rozumie się każdą funkcję przedziałami wielomianową. Zadanie interpolacyjne polega zatem na znalezieniu w ustalonej klasie funkcji interpolującej. Przebieg wykresu interpolacyjnego można przedstawić w sposób graficzny. Pewna funkcja dyskretna $y = f(x)$ została zaprezentowana na rys. 2a. Na rys. 2b pokazano jedną z możliwych interpretracji funkcji $f(x)$ w przestrzeni funkcyjnej. Jest to interpolacja za pomocą wielomianu $y = -\frac{1}{3}x^3 + x^2 + \frac{1}{3}x$. Oczywiście można zaproponować także inne interpolacje – na przykład interpolację za pomocą wielomianu trygonometrycznego, gdzie zastosowanie znajduje szereg Fouriera. Uzyskany wielomian trygonometryczny jest zespolony, mimo że interpolowana funkcja może być rzeczywista. Będzie o tym mowa w dalszej części tej książki.

Zakładając, że sygnał nie zmienia się między pomiarami, funkcja może zostać przedłużona (uzupełniona) do funkcji odcinkowo-stałej (rys. 2c). Oznacza to, że przedłużona funkcja $\Gamma(x)$ jest stała w każdym z podprzedziałów $[n, (n+1))$, $n = 0, 1, ...$ Jest to więc inna reprezentacja tej samej funkcji $f(x)$. Funkcja $\Gamma(x)$ jest określona dla wszystkich x i jest ciągła wszędzie, poza wartościami całkowitymi x. W punktach $0, 1, 2, ...$ funkcja jest ciągła z prawej strony i nieciągła z lewej.



Rys. 2. Funkcja dyskretna (a) i jej odpowiednik w postaci funkcji interpolującej (b) oraz odcinkowo-stałej (c). Kółko oznacza izolowany punkt nienależący do krzywej, kropka – punkt należący do krzywej

5. Wybrane dyskretne transformacje i transformaty

Jednym z ważnych powodów transformowania posiadanych danych z jednej postaci w drugą jest obserwacja, że pewne zależności między danymi są lepiej widoczne po ich transformacji niż podczas ich obserwacji w postaci oryginalnej (pierwotnej). Takie transformacje nazywane są również przekształceniami. Ponieważ przekształcenia są szczególnym przypadkiem odwzorowań, możemy również mówić o odwzorowaniach. Ważną grupę przekształceń stanowią przekształcenia liniowe. Przekształcenie lub odwzorowanie liniowe jest jednym z aksjomatów algebry liniowej. Jest to odwzorowanie homomorficzne, zachowujące strukturę przestrzeni, a więc działania dodawania wektorów i mnożenia przez skalar.

Szczególnym przypadkiem przekształceń liniowych są przekształcenia ortogonalne – reprezentowane przez odpowiednie macierze ortogonalne. Charakterystyczną własnością macierzy ortogonalnej jest to, że jej odwrotność jest tym samym, co jej transpozycja, o czym była już mowa w rozdziale 1. Własność ta jest niezwykle użyteczna. W dwóch wymiarach łatwo jest obliczyć odwrotność macierzy, ale ze wzrostem liczby wymiarów takie obliczenia stają się bardzo uciążliwe i kosztowne obliczeniowo. Transpozycja jest operacją bardzo prostą. Z tego względu obecność macierzy ortogonalnej stanowi udogodnienie obliczeniowe.

Przekształcenie liniowe może być określone przez działanie na dowolnej bazie, nie ma więc znaczenia, jaka baza zostanie zastosowana. Zasadę tę można oczywiście rozszerzyć na dowolną liczbę wymiarów przestrzeni – dowolna baza przestrzeni n wymiarowej zawiera zawsze n niezależnych wektorów. Należy zwrócić uwagę, że jedne bazy mogą być bardzo wygodne w obliczeniach i dawać wyobrażenie, jak przebiega samo przekształcenie. Inne bazy prowadzą natomiast do skomplikowanych obliczeń, a otrzymywane wyniki są trudne i mało intuicyjne w interpretacji. Przekształcenia liniowe badamy za pomocą ich macierzy w pewnych bazach, dlatego chcemy tak dobierać bazy, aby macierze te miały możliwie prostą budowę.

Jak wiadomo, każde odwzorowanie może być określone przez podanie formuły, jak wektory bazy oddziałują na dane pierwotne.

Tak więc można interpretować transformację jako sposób dekompozycji zebranych próbek danych na sumę wektorów bazowych.

Znanych jest dużo systemów tworzących zbiory funkcji bazowych. Funkcje te mają wiele ciekawych własności. Wszystkie znane funkcje bazowe spełniają postulaty sformułowane po raz pierwszy przez Josepha Fouriera (1768–1830), francuskiego matematyka, którego uważa się za prekursora współczesnej analizy harmonicznej. Powszechne stosowanie funkcji harmonicznych jako funkcji bazowych wiąże się przede wszystkim ze znanym faktem, że sygnały harmoniczne są jedynymi sygnałami, które po przejściu przez układ liniowy nie zmieniają swojego kształtu – nadal są funkcjami harmonicznymi. Zmianom mogą podlegać jedynie faza i amplituda funkcji harmonicznych [9, 40].

W wielu obszarach praktycznych zastosowań własność ta może być ignorowana, gdyż w praktyce inżynierskiej istotne są często inne własności funkcji bazowych uzasadniające zastąpienie funkcji harmonicznych innymi, bardziej użytecznymi z punktu widzenia zastosowań, funkcjami bazowymi. Będzie o tym mowa w kolejnych rozdziałach.

5.1. Transformata (widmo)

W niniejszym rozdziale zaprezentowano modele teoretyczne użytecznych znanych i mniej znanych transformacji. Na kanwie wywodów teoretycznych przedstawiono algorytmy, które umożliwiają realizację transformacji dla różnych danych wejściowych. Pozwala to na głębsze poznanie mechanizmów ukrytych w niuansach teorii. Oprócz algorytmów zapisanych w języku Matlab, pokazano także liczne przykłady rachunkowe.

Terminologicznie przyjęto odróżniać transformację – operację wykonywaną na danych pierwotnych – od transformaty – wyniku tej operacji. Transformata nazywana jest widmem sygnału pierwotnego. W rozdziale tym interesować nas będą wyłącznie dyskretne reprezentacje sygnałów, a więc ciągi liczb (albo wektory) – te jak wiadomo są elementami przestrzeni wektorowej.

Niech $\boldsymbol{E}^n$ jest n wymiarową przestrzenią wektorową rozpiętą na wektorach ortonormalnej bazy $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n\}$. Każdy wektor (np. wektor $\boldsymbol{b}$) należący do tej przestrzeni można przedstawić w sposób jednoznaczny za pomocą liniowej kombinacji wektorów bazy:

$$\boldsymbol{b} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i, \tag{5.1}$$

gdzie $\boldsymbol{b} = [b_1, b_2, ..., b_n]$.

Współczynniki α_i wyznacza się w ten sposób, że tworzone są iloczyny skalarne obu stron równania (5.1) z wektorami bazowymi $\boldsymbol{v}_j$, $j = 1, 2, ..., n$:

$$\langle \boldsymbol{b}, \boldsymbol{v}_j \rangle = \left\langle \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i, \boldsymbol{v}_j \right\rangle. \tag{5.2}$$

Z własności iloczynu skalarnego wynika, że $\langle \alpha \boldsymbol{d}, \boldsymbol{e} \rangle = \alpha \langle \boldsymbol{d}, \boldsymbol{e} \rangle$ dla dowolnych $\boldsymbol{d}, \boldsymbol{e} \in \boldsymbol{V}$ oraz dla każdego $\alpha \in \boldsymbol{R}$, a więc:

$$\langle \boldsymbol{b}, \boldsymbol{v}_j \rangle = \sum_{i=1}^{n} \alpha_i \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle. \tag{5.3}$$

W ten sposób można zbudować układ n równań liniowych z n niewiadomymi α_i. Układ tych równań można zapisać w zwartej formie macierzowej:

$$\boldsymbol{V} \cdot \boldsymbol{\alpha} = \boldsymbol{c}, \tag{5.4}$$

gdzie:

$$\boldsymbol{V} = \begin{bmatrix} \langle \boldsymbol{v}_1, \boldsymbol{v}_1 \rangle & \cdots & \langle \boldsymbol{v}_1, \boldsymbol{v}_n \rangle \\ \vdots & \ddots & \vdots \\ \langle \boldsymbol{v}_n, \boldsymbol{v}_1 \rangle & \cdots & \langle \boldsymbol{v}_n, \boldsymbol{v}_n \rangle \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} \langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle \\ \vdots \\ \langle \boldsymbol{b}, \boldsymbol{v}_n \rangle \end{bmatrix}. \tag{5.5}$$

Wartości współczynników można obliczyć ze wzoru:

$$\boldsymbol{\alpha} = \boldsymbol{V}^{-1} \cdot \boldsymbol{c}. \tag{5.6}$$

Baza przestrzeni $\boldsymbol{E}^n$ jest ortonormalna, dlatego wektory $\boldsymbol{v}_i$ są unormowane, a to z kolei oznacza, że występujące we wzorze (5.5) iloczyny skalarne przyjmują tylko dwie wartości: $\langle \boldsymbol{v}_i, \boldsymbol{v}_i \rangle = 1$ i $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = 0$. Macierz $\boldsymbol{V}$ jest więc macierzą jednostkową. Zatem macierz odwrotna do niej jest także macierzą jednostkową. Równanie (5.6) można więc zapisać inaczej:

$$\boldsymbol{\alpha} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \cdot \boldsymbol{c}. \tag{5.7}$$

Oznacza to, że współczynniki α_i wyznaczyć można bezpośrednio z iloczynu skalarnego wektora $\boldsymbol{b}$ oraz odpowiedniego wektora bazowego $\boldsymbol{v}_i$:

$$\alpha_i = \langle \boldsymbol{b}, \boldsymbol{v}_i \rangle \qquad \text{dla} \quad i = 1, 2, ..., N. \tag{5.8}$$

Współczynniki α_i są współrzędnymi wektora $\boldsymbol{b}$ w układzie współrzędnych wyznaczonym przez wektory bazy $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n\}$ przestrzeni $\boldsymbol{E}^n$.

Jeśli liniową euklidesową przestrzeń wektorową $\boldsymbol{E}^n$ z bazą ortonormalną $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n\}$ zastąpić liniową przestrzenią euklidesową $\boldsymbol{E}'^n$ z odpowiednio dobraną bazą ortogonalną $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n\}$, to dowolny wektor tej przestrzeni można przedstawić w postaci kombinacji liniowej wektorów $\boldsymbol{u}_i$, $i = 1, 2, ..., n$:

$$\boldsymbol{b} = \sum_{i=1}^{n} \alpha_i' \boldsymbol{u}_i. \tag{5.9}$$

Tworząc iloczyny skalarne, dla obydwu stron równania (5.9) otrzymujemy:

$$\langle \boldsymbol{b}, \boldsymbol{u}_j \rangle = \sum_{i=1}^{n} \alpha_i' \langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle . \tag{5.10}$$

Postępując podobnie jak poprzednio, można zbudować macierz $\boldsymbol{V}$. Sposób budowania takiej macierzy dla różnych przypadków został szczegółowo przedstawiony w tym rozdziale.

Ponieważ baza jest ortogonalna, macierz $\boldsymbol{V}$ jest macierzą diagonalną typu:

$$\boldsymbol{V} = \begin{bmatrix} \lambda & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda \end{bmatrix} \quad \text{oraz} \quad \boldsymbol{V}^{-1} = \begin{bmatrix} \frac{1}{\lambda} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\lambda} \end{bmatrix}, \tag{5.11}$$

gdzie $\lambda = \|\boldsymbol{u}_i\|^2$. Wartości współczynników można więc obliczyć z wzoru:

$$\boldsymbol{\alpha}' = \boldsymbol{V}^{-1} \cdot \boldsymbol{c} = \frac{1}{\lambda} \cdot \boldsymbol{c}, \qquad \text{gdzie } \boldsymbol{c} = \begin{bmatrix} \langle \boldsymbol{b}, \boldsymbol{u}_1 \rangle \\ \vdots \\ \langle \boldsymbol{b}, \boldsymbol{u}_n \rangle \end{bmatrix}. \tag{5.12}$$

Poszczególne współczynniki α'_i można wyznaczyć z równania:

$$\alpha'_i = \frac{1}{\lambda} \cdot \langle \boldsymbol{b}, \boldsymbol{u}_i \rangle, \quad \text{dla} \quad i = 1, 2, ..., n. \tag{5.13}$$

Wektory bazy $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n\}$ są ortogonalne, ale mogą mieć, inaczej niż powyżej, różne normy (są różnej długości), wtedy $\lambda_i = \|\boldsymbol{u}_i\|^2$. W takim przypadku wzór (5.13) podlega prostej modyfikacji:

$$\alpha'_i = \frac{1}{\lambda_i} \cdot \langle \boldsymbol{b}, \boldsymbol{u}_i \rangle, \quad \text{dla} \quad i = 1, 2, ..., n. \tag{5.14}$$

Przykład 5.1. *Niech $\boldsymbol{E}^4$ jest ortonormalną euklidesową przestrzenią wektorową rozpiętą na wektorach bazy $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4\}$:*
$\boldsymbol{v}_1 = [\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$, $\boldsymbol{v}_2 = [\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}]$, $\boldsymbol{v}_3 = [\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}]$, $\boldsymbol{v}_4 = [\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}]$.
Należy znaleźć reprezentację wektora $\boldsymbol{b} = [1, 2, 3, 4]$ w przestrzeni $\boldsymbol{E}^4$.
Układając wierszami wektory bazowe, otrzymamy macierz:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \\ \boldsymbol{v}_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad \textit{oraz} \quad \boldsymbol{A}^{-1} = \boldsymbol{A}.$$

Macierz $\boldsymbol{A}$ *jest macierzą ortogonalną i symetryczną, gdyż* $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$ *oraz* $\boldsymbol{A}^T = \boldsymbol{A}$, $det(\boldsymbol{A}) = 1$. *Wektory bazowe* $\boldsymbol{v}_i$ *są unormowane, gdyż* $\|\boldsymbol{v}_i\| = 1$, *tak więc* $\lambda_i = 1$ *dla* $i = 1, 2, ..., 4$. *Zgodnie z* (5.1) *oraz* (5.8), *wartości współczynników* α_i, *które są współrzędnymi wektora* $\boldsymbol{b}$ *w układzie współrzędnych wyznaczonym przez bazę* $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4\}$, *są następujące*:

$$\begin{bmatrix} \langle \boldsymbol{b}, \boldsymbol{v}_1 \rangle \\ \langle \boldsymbol{b}, \boldsymbol{v}_2 \rangle \\ \langle \boldsymbol{b}, \boldsymbol{v}_3 \rangle \\ \langle \boldsymbol{b}, \boldsymbol{v}_4 \rangle \end{bmatrix} = \boldsymbol{b} \cdot \boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} =$$
$$= \begin{bmatrix} 5 & -1 & -2 & 0 \end{bmatrix}.$$

Wektor $\boldsymbol{b}$ *można jednoznacznie przedstawić w postaci kombinacji liniowej wektorów bazy* $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4\}$ *w sposób następujący*:

$$\boldsymbol{b} = [1, 2, 3, 4] = 5 \cdot \boldsymbol{v}_1 - 1 \cdot \boldsymbol{v}_2 - 2 \cdot \boldsymbol{v}_3 + 0 \cdot \boldsymbol{v}_4 =$$
$$= 5 \cdot [\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}] - 1 \cdot [\tfrac{1}{2}, -\tfrac{1}{2}, \tfrac{1}{2}, -\tfrac{1}{2}] - 2 \cdot [\tfrac{1}{2}, \tfrac{1}{2}, -\tfrac{1}{2}, -\tfrac{1}{2}].$$

Jeśli baza $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n\}$ *przestrzeni* $\boldsymbol{E}^4$*jest ortogonalna, a wektory mają postać* $\boldsymbol{u}_1 = [1, 1, 1, 1]$, $\boldsymbol{u}_2 = [1, -1, 1, -1]$, $\boldsymbol{u}_3 = [1, 1, -1, -1]$, $\boldsymbol{u}_4 = [1, -1, -1, 1]$, *z normą* $\|u_i\| = 2$, *dla* $i = 1, 2, ..., n$, *to macierz* $\boldsymbol{B}$ *zbudowana z tych wektorów ma postać*:

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \boldsymbol{v}_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix},$$

$$\boldsymbol{B}^{-1} = \frac{1}{4} \cdot \boldsymbol{B} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix},$$

$\boldsymbol{B}^{-1} \neq \boldsymbol{B}^T$ *oraz* $\boldsymbol{B}^T = \boldsymbol{B}$ *i* $det(\boldsymbol{B}) = 16$. *Wówczas na podstawie* (5.13):

$$\frac{1}{\lambda} \cdot \begin{bmatrix} \langle \boldsymbol{b}, \boldsymbol{u}_1 \rangle \\ \langle \boldsymbol{b}, \boldsymbol{u}_2 \rangle \\ \langle \boldsymbol{b}, \boldsymbol{u}_3 \rangle \\ \langle \boldsymbol{b}, \boldsymbol{u}_4 \rangle \end{bmatrix} = \frac{1}{\lambda} \cdot \boldsymbol{b} \cdot \boldsymbol{B} = \frac{1}{4} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{10}{4} & -\frac{2}{4} & -1 & 0 \end{bmatrix}.$$

Wektor $\boldsymbol{b}$ *można więc jednoznacznie przedstawić w postaci kombinacji liniowej wektorów bazy ortogonalnej* $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{u}_4\}$*w sposób następujący*:

$$\begin{aligned} \boldsymbol{b} = [1,2,3,4] &= \tfrac{10}{4} \cdot \boldsymbol{u}_1 - \tfrac{2}{4} \cdot \boldsymbol{u}_2 - 1 \cdot \boldsymbol{u}_3 + 0 \cdot \boldsymbol{u}_4 = \\ &= \tfrac{10}{4} \cdot [1,1,1,1] - \tfrac{2}{4} \cdot [1,-1,1,-1] - 1 \cdot [1,1,-1,-1]. \end{aligned}$$

Przykład 5.2. *Niech przestrzeń* $\boldsymbol{E}^3$ *jest przestrzenią wektorową rozpiętą na wektorach bazy* $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$.
Niech $\boldsymbol{w}_1 = [1,2,3]$, $\boldsymbol{w}_2 = [\frac{12}{7}, \frac{3}{7}, -\frac{6}{7}]$, $\boldsymbol{w}_3 = [\frac{1}{2}, -1, \frac{1}{2}]$. *Baza jest bazą ortogonalną. Należy znaleźć reprezentację wektora* $\boldsymbol{b} = [2,-2,3]$ *w postaci kombinacji liniowej wektorów bazy. Postępując podobnie jak w poprzednim przykładzie, otrzymujemy*:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \boldsymbol{w}_3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ \frac{12}{7} & \frac{3}{7} & -\frac{6}{7} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{bmatrix}.$$

Wyznaczamy także: $\lambda_1 = \|\boldsymbol{w}_1\|^2 = 14$, $\lambda_2 = ||\boldsymbol{w}_2||^2 = \frac{27}{7}$, $\lambda_3 = \|\boldsymbol{w}_3\|^2 = \frac{3}{2}$, *co oznacza, że wektory mają różne długości. Zgodnie ze wzorem* (5.14), *otrzymujemy*:

$$\begin{aligned} \alpha'_1 &= \tfrac{1}{\lambda_1} \cdot \langle \boldsymbol{b}, \boldsymbol{w}_1 \rangle = \tfrac{1}{14} \cdot ([2,-2,3] \circ [1,2,3]) = \tfrac{1}{2}, \\ \alpha'_2 &= \tfrac{1}{\lambda_2} \cdot \langle \boldsymbol{b}, \boldsymbol{w}_2 \rangle = \tfrac{7}{27} \cdot \left([2,-2,3] \circ [\tfrac{12}{7}, \tfrac{3}{7}, -\tfrac{6}{7}]\right) = 0, \\ \alpha'_3 &= \tfrac{1}{\lambda_3} \cdot \langle \boldsymbol{b}, \boldsymbol{w}_3 \rangle = \tfrac{2}{3} \cdot \left([2,-2,3] \circ [\tfrac{1}{2}, -1, \tfrac{1}{2}]\right) = 3. \end{aligned}$$

Jak widać, wyznaczenie współczynników α_i' nie wymaga bezpośredniego odwracania macierzy $\boldsymbol{A}$, chociaż formalnie ta operacja jest ukryta we wzorach (5.2). Wykonując jawnie tę operację, otrzymujemy identyczne wartości współczynników widmowych:

$$\boldsymbol{b} \cdot \boldsymbol{A}^{-1} = \begin{bmatrix} 2 & -2 & 3 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{14} & \frac{4}{9} & \frac{1}{3} \\ \frac{1}{7} & \frac{1}{9} & -\frac{2}{3} \\ \frac{3}{14} & -\frac{2}{9} & \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 3 \end{bmatrix}. \tag{5.15}$$

Wektor $\boldsymbol{b}$ można zatem jednoznacznie rozwinąć względem wektorów bazowych $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$ w sposób następujący:

$$\boldsymbol{b} = [2, -2, 3] = \tfrac{1}{2} \cdot \boldsymbol{w}_1 + 0 \cdot \boldsymbol{w}_2 + 3 \cdot \boldsymbol{w}_3 = \tfrac{1}{2} \cdot [1, 2, 3] + 3 \cdot [\tfrac{1}{2}, -1, \tfrac{1}{2}].$$

Klasyczne, znane z podstaw algebry liniowej, sposoby odwracania macierzy są z reguły uciążliwe i kosztowne obliczeniowo [16, 31]. Odpowiednio dobrane bazy pozwalają jednak na znaczne uproszczenie obliczeń, co ilustrują przytoczone wcześniej dwa przykłady. Wyznaczane współczynniki α_i (α_i') nazywane są współczynnikami widmowymi, a ich zbiór $\{\alpha_i\}_{i=0}^n$ lub $\{\alpha_i'\}_{i=0}^n$ – widmem. Często mówi się o współczynnikach Fouriera w konkretnej bazie, np. w bazie Walsha, Hartleya, Haara itp. Przedstawione powyżej rozważania można również odnieść do funkcji ciągłych, w szczególności do funkcji o wyjątkowo skomplikowanym zapisie. Upraszczamy ten zapis funkcją interpolującą, która jest skończonym szeregiem Fouriera w wybranej bazie. Interpolacja wielomianami trygonometrycznymi występuje często w przypadku funkcji znanych z danych pomiarowych, co można utożsamiać z pobraniem próbek wartości pewnego sygnału analogowego. Wyznaczenie współczynników poszukiwanego wielomianu trygonometrycznego stopnia N wymaga wtedy obliczeń N^2 wartości funkcji trygonometrycznych. Niektóre wartości pokrywają się po zastosowaniu tożsamości trygonometrycznych, co w 1965 r. zauważyli J. Cooley i J.W. Tukey [12], obniżając znacząco złożoność obliczeniową wyznaczania współczynników poszukiwanego wielomianu. Ponieważ szereg jest skończony, ma skończoną liczbę wyrazów, interpolacja nie odtwarza dokładnie przebiegu pierwotnego.

Z oczywistych względów poszukuje się najlepszej interpolacji, co jest równoznaczne z poszukiwaniem baz, dla których błędy interpolacji są najmniejsze. Same transformacje mogą być zarówno jedno-, jak i wielowymiarowe. Będzie o tym mowa w kolejnych rozdziałach książki.

5.2. Dyskretna transformacja Fouriera

W dziedzinie przetwarzania sygnałów transformacja Fouriera jest bez najmniejszej wątpliwości narzędziem fundamentalnym, znanym i stosowanym od dawna w różnych dziedzinach nauki i techniki. Transformata Fouriera jest ważnym narzędziem analizy harmonicznej oraz przetwarzania sygnałów. Może być stosowana zarówno dla sygnałów ciągłych, jak i dyskretnych lub cyfrowych. W swoich założeniach transfomacja Fouriera związana jest ściśle z pojęciem częstotliwości i znajduje zastosowanie w dziedzinach, w których analiza częstotliwościowa ma duże znaczenie dla obserwatora-badacza, np.: w akustyce, wibroakustyce, kompresji sygnałów, analizie sygnałów harmonicznych, projektowaniu filtrów przeciwzakłóceniowych itp. Często jednak informacje na temat częstotliwości są mało użyteczne, a interpretacja wyników niepotrzebnie skomplikowana. Można wtedy stosować inne transformacje. Wiele przekształceń wzoruje się jednak na idei transformacji Fouriera.

Wykonywanie obliczeń za pomocą komputera wymaga, aby pierwotny sygnał analogowy posiadał reprezentację cyfrową. Odbywa się to w przetworniku analogowo-cyfrowym (A/C), który zastępuje zmieniający się płynnie sygnał analogowy, z ustalonym poziomem dokładności, sygnałem zmieniającym się skokowo. W ten sposób uzyskujemy uproszczony sygnał analogowy – sygnał cyfrowy. Sygnał cyfrowy można zapisać w postaci zero-jedynkowej lub dziesiętnej. Rejestrujemy więc N elementowy ciąg liczb wymiernych, które odpowiadają wybranym punktom sygnału analogowego, co ilustruje rys. 1.

Transformacja będzie miała również sens, gdy obliczenia przeprowadzone będą dla dowolnego, uporządkowanego ciągu liczb. Z takimi przykładami będziemy jeszcze mieli do czynienia.

5.2.1. Jednowymiarowa transformacja Fouriera

Definicja 5.1. *Jednowymiarową dyskretną transformacją Fouriera (ang. Discrete Fourier Transform – DFT) sygnału $x(n)$, określonego w chwilach $n = 0, 1, ..., N-1$, nazywamy ciąg zespolonych współczynników rozwinięcia sygnału $x(n)$ [34]:*

$$s(k) = \alpha \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{n}{N} k}, \quad k = 0, 1, ..., N-1, \tag{5.16}$$

gdzie $e^{-j\frac{2\pi}{N}}$ jest głównym pierwiastkiem N-tego stopnia z j.

Wynikiem transformacji (5.16) jest transformata (widmo) $s(k)$.

Definicja 5.2. *Jednowymiarową dyskretną odwrotną transformacją Fouriera (ang. Inverse Discrete Fourier Transform – IDFT) jest odwzorowanie, które na podstawie widma sygnału $s(k)$, $k = 0, ..., N-1$, odtwarza próbki sygnału pierwotnego $x(n)$, $n = 0, ..., N-1$:*

$$x(n) = \beta \sum_{k=0}^{N-1} s(k) e^{j2\pi \frac{n}{N} k}, \quad n = 0, 1, ..., N-1. \tag{5.17}$$

Funkcje $e^{-j2\pi \frac{n}{N} k}$ są funkcjami okresowymi o okresie N:

$$e^{-j\frac{2\pi}{N}(k+N)n} = e^{-j\frac{2\pi}{N}kn} + e^{-j2\pi n} = e^{-j\frac{2\pi}{N}kn}. \tag{5.18}$$

Wynika stąd, że widma sygnałów dyskretnych są okresowe. Okresowość widma jest cechą sygnałów dyskretnych. Dyskretną transformację Fouriera możemy traktować jako unitarne przekształcenie wektora próbek w wektor składowych fourierowskich. DFT można zatem traktować jako przedstawienie wektora próbek danych w bazie funkcji trygonometrycznych. W tym celu wystarczy skorzystać z tożsamości Eulera, która wiąże funkcje trygonometryczne $\sin(x)$ oraz $\cos(x)$ z zespoloną funkcją wykładniczą, występującą we wzorach definicyjnych prostej i odwrotnej transformacji Fouriera. Te ważne zależności będą szczegółowo omówione.

Ponieważ $\cos(-x) = \cos(x)$ oraz $\sin(-x) = -\sin(x)$, zatem:

$$\begin{aligned} &e^{jx} = \cos(x) + j\sin(x), \\ &e^{-jx} = e^{j(-x)} = \cos(-x) + j\sin(-x) = \cos(x) - j\sin(x). \end{aligned} \tag{5.19}$$

Wzór (5.16) wskazuje, że bazę transfomacji Fouriera stanowią funkcje trygonometryczne. Jest to baza ortogonalna. Transformacje (5.16) oraz (5.17) będą wzajemnie odwrotne, jeśli iloczyn współczynników α i β spełni zależność:

$$\alpha \cdot \beta = \frac{1}{N}. \tag{5.20}$$

Ze względów praktycznych najczęściej stosowane są następujące pary współczynników $(\alpha, \beta) : (\frac{1}{N}, 1), (1, \frac{1}{N}), (\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}})$. Wybór pary $(\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}})$ sprawia, że transformacja Fouriera staje się transformacją unitarną. Ma to jednak znaczenie wyłącznie teoretyczne.

Transformacja Fouriera produkuje transformatę, której współczynniki są w ogólności liczbami zespolonymi. Z tego powodu część rzeczywista i urojona transformaty są z sobą powiązane. Można więc określić dodatkowe widma – widmo amplitudowe i fazowe. Widmo amplitudowe (wykres modułu) pozwala określić, jakie są amplitudy składowych widma sygnału o różnych częstotliwościach. Widmo fazowe (wykres argumentu) pokazuje, jakie są fazy tych składowych.

Definicja 5.3. *Widmo amplitudowe sygnału $x(n)$ jest ciągiem liczb rzeczywistych będących modułami współczynników $s(k)$ rozwinięcia sygnału $x(n)$, $n = 0, ..., N-1$:*

$$|s(k)| = \sqrt{Re[s(k)]^2 + Im[s(k)]^2}, k = 0, ..., N-1, \tag{5.21}$$

gdzie:
$Re[s(k)]$ oraz $Im[s(k)]$ oznaczają odpowiednio część rzeczywistą i urojoną liczby $s(k)$.

Definicja 5.4. *Widmo fazowe sygnału $x(n)$ jest ciągiem rzeczywistym współczynników faz* $\theta = \arctan\left(\frac{Im(s(k))}{Re(s(k))}\right)$. *Jest to więc ciąg modułów współczynników $s(k)$. Aby uniknąć przypadków dzielenia przez zero, stosuje się następujący zabieg.*
Niech $x = Re[s(k)]$ oraz $y = Im[s(k)]$, wtedy:

$$\theta_k = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \pi + \arctan\left(\frac{y}{x}\right) & y \geq 0,\ x < 0 \\ -\pi + \arctan\left(\frac{y}{x}\right) & y < 0,\ x < 0 \\ \pi/2 & y > 0,\ x = 0 \\ -\pi/2 & y < 0,\ x = 0 \end{cases}, \ k = 0, ..., N-1. \tag{5.22}$$

Wzór (5.22) informuje, że argument θ liczby zespolonej jest tzw. argumentem głównym tej liczby. Wartość argumentu głównego mieści się w przedziale $-\pi < \theta \leq \pi$.

W Matlabie zależność (5.22) realizuje polecenie **atan2**(y,x) lub **angle(**$x + jy$**)**. Często stosuje się dodatkową procedurę, polegającą na takim doborze kątów θ, aby ich wartości zmieniały się monotonicznie. W Matlabie realizuje tę zasadę polecenie **unwrap**(p), gdzie p jest ciągiem wartości kątów (w radianach), które należy odpowiednio dobrać przez dodanie do wybranych z nich wartości $\pm 2\pi$. Zawsze spełnione jest równanie:

$$s(k) = |s(k)|e^{j\theta_k}, \quad k = 0, ..., N-1. \tag{5.23}$$

Przykład 5.3. *Znajdowanie współczynników widmowych dyskretnej transformacji Fouriera wektora $\boldsymbol{x} = [1, 2, -2, 0]$. Współrzędne tego wektora mogą być oczywiście utożsamiane z próbkami $x(0) = 1$, $x(1) = 2$, $x(2) = -2$, $x(3) = 0$ pewnego rzeczywistego, zarejestrowanego sygnału analogowego. Przyjęto, że $\alpha = 1$. Na podstawie (5.16) można obliczyć poszczególne współczynniki widmowe:*

$$s(0) = \sum_{n=0}^{3} x(n)e^{0} = x(0) + x(1) + x(2) + x(3) = 1 + 2 - 2 + 0 = 1,$$

$$s(1) = \sum_{n=0}^{3} x(n)e^{-j2\pi n/4} = 1e^{0} + 2e^{-j\pi/2} - 2e^{-j\pi} + 0e^{-j6\pi/4} = 3 - 2j,$$

$$s(2) = \sum_{n=0}^{3} x(n)e^{-j\pi n} = 1e^{0} + 2e^{-j\pi} - 2e^{-j2\pi} + 0e^{-j3\pi} = -3,$$

$$s(3) = \sum_{n=0}^{3} x(n)e^{-j6\pi n/4} = 1e^{0} + 2e^{-j3\pi/2} - 2e^{-j3\pi} + 0e^{-j9\pi/4} = 3 + 2j.$$

Ciąg współczynników widma amplitudowego określony jest następującymi wartościami: $|s(0)| = 1$, $|s(1)| = \sqrt{3^2 + (-2)^2} = \sqrt{13}$, $|s(2)| = 3$, $|s(3)| = \sqrt{13}$. *Współczynniki faz określone są następującym ciągiem monotonicznych zmian wartości*: $\theta_0 = 0$, $\theta_1 = -0.5880$, $\theta_2 = -\pi$, $\theta_3 = -5.6952$.

Wykonując podobne rachunki, można na podstawie widma odtworzyć elementy wektora $\boldsymbol{x}$. Mamy wtedy do czynienia z transformacją odwrotną (IDFT). Wykorzystuje się do tego celu zależność (5.17). Dla $\alpha = 1$ należy przyjąć $\beta = \frac{1}{N}$. Można zauważyć, że przekształcenie (5.16) można zapisać w równoważnej, macierzowej postaci:

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{x} \cdot \boldsymbol{W}_a, \tag{5.24}$$

gdzie $\boldsymbol{W}_a = [w_{m,k}]_{N\times N} = [e^{-j\frac{2\pi}{N}mk}]$.

Przekształcenie (5.17) będzie miało postać macierzową:

$$\boldsymbol{x} = \beta \cdot \boldsymbol{s} \cdot \boldsymbol{W}_b, \tag{5.25}$$

gdzie $\boldsymbol{W}_b = [w_{n,k}]_{N\times N} = [e^{j\frac{2\pi}{N}nk}]$.

Zamiast wzorów (5.24) i (5.25) można prowadzać obliczenia inaczej, co wynika z własności macierzy $\boldsymbol{W}_a$ i $\boldsymbol{W}_b$, gdyż odpowiadające sobie elementy tych

macierzy są sprzężone: $\boldsymbol{W} = \boldsymbol{W}_a = \overline{\boldsymbol{W}_b}$ oraz $\boldsymbol{W}^{-1} = \overline{\boldsymbol{W}}$, wtedy:

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{x} \cdot \boldsymbol{W} \tag{5.26}$$

oraz

$$\boldsymbol{x} = \beta \cdot \boldsymbol{s} \cdot \overline{\boldsymbol{W}}, \tag{5.27}$$

gdzie $\boldsymbol{s}, \boldsymbol{x} \in \boldsymbol{C}^N$. Macierz $\boldsymbol{W} \in \boldsymbol{C}^{N \times N}$ może być dowolną macierzą typu $\boldsymbol{W}_a$ lub $\boldsymbol{W}_b$, a $\overline{\boldsymbol{W}}$ jest macierzą sprzężoną z macierzą $\boldsymbol{W}$ oraz $\boldsymbol{W} = \boldsymbol{W}^T$.

Macierz sprzężoną otrzymamy, odwracając na przeciwne znaki wszystkich urojonych elementów macierzy $\boldsymbol{W}$ – zasada wzajemnej odwrotności związków (5.26) i (5.27) jest także automatycznie zachowana. Uwzględniając własności wymienionych macierzy, zapis pary transformat można przedstawić w postaci:

$$\boldsymbol{s} = \boldsymbol{x} \cdot \boldsymbol{W}, \tag{5.28}$$

$$\boldsymbol{x} = \boldsymbol{s} \cdot \boldsymbol{W}^{-1}. \tag{5.29}$$

Operacja odwracania macierzy jest kosztowna obliczeniowo i w realizacjach praktycznych, szczególnie w przypadku dużych rozmiarów macierzy $\boldsymbol{W}$, nie jest stosowana. Z przedstawionych rozważań wynika, że metoda bezpośrednia, zgodnie ze wzorem (5.24), wymaga mnożenia macierzy $\boldsymbol{W}$ o rozmiarach $N \times N$ przez N wymiarowy wektor $\boldsymbol{x}$ (naprawdę przez wektor transponowany $\boldsymbol{x}^T$). Realizacja tego zadania wymaga N^2 zespolonych mnożeń oraz $N \cdot (N-1)$ zespolonych dodawań. DFT jest więc, z dokładnością do czynnika normalizującego, unitarnym przekształceniem wektora próbek na wektor widma – składowych fourierowskich. Ponieważ mamy do czynienia z procesem dyskretnym, współrzędne wektora $\boldsymbol{x}$ można utożsamiać z wartościami pewnej funkcji $f(t)$, określonej w punktach przedziału $[0, N-1]$. Realizacja wyrażonej macierzowo transformacji (5.24) ma więc złożoność $O(N^2)$. Ze względu na wymagany czas obliczeń w aplikacjach czasu rzeczywistego wykonanie takiego zadania jest utrudnione, a dla dużych N niemożliwe.

Przykład 5.4. *Wyznaczanie współczynników widmowych Fouriera wektora danych dyskretnych* $\boldsymbol{x} = [1, 2, -2, 0]$. *Przyjmując* $\alpha = 1$, $\beta = 1/4$, *otrzymujemy*:

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{x} \cdot \boldsymbol{W} = \begin{bmatrix} 1 & 2 & -2 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 3-2j & -3 & 3+2j \end{bmatrix}.$$

Odwracanie macierzy $\boldsymbol{W}$ *można sprowadzić do wyznaczenia macierzy sprzężonej* $\overline{\boldsymbol{W}}$:

$$\overline{\boldsymbol{W}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}.$$

Znając widmo sygnału dyskretnego, można łatwo odtworzyć oryginalne wartości próbek sygnału:

$$\boldsymbol{x} = \beta \cdot \boldsymbol{s} \cdot \overline{\boldsymbol{W}} = \frac{1}{4} \cdot \begin{bmatrix} 1 & 3-2j & -3 & 3+2j \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 2 & -2 & 0 \end{bmatrix}.$$

Opisane powyżej metody wyznaczania prostej i odwrotnej dyskretnej transformacji Fouriera zaprogramowane są, w postaci odpowienich funkcji lub procedur, w wielu pakietach matematycznych, np. Maple lub Matlab. Obliczenia te zaprogramować można w różny sposób, uzyskując te same wyniki. Jeśli to konieczne, to należy zwrócić uwagę na precyzję i czas wykonania obliczeń, które mogą być różne dla różnych sposobów programowania zagadnienia.

Różnice precyzji obliczeń można zaobserwować, posługując się na przykład oprogramowaniem Matlab.

Program 5.1. *Program, za pomocą którego można w Matlabie wyznaczyć trzema sposobami jednowymiarowe widmo Fouriera wektora* $\boldsymbol{x}$.

```
x = 1:16;          % Liczba probek sygnalu(wektor danych)
N = length(x);     % Liczba elementow wektora wejsciowego
a = fft(x);        % 1 SPOSOB. Za pomoca funkcji Matlaba fft()
b = x*dftmtx(N);   % 2 SPOSOB. Za pomoca funkcji Matlaba dftmtx()
p = norm(a-b)      % Roznica dlugosci wektorow wsp. widmowych
a4 = vpa(a(4))         % Wartosc czwartego wsp. widmowego
b4 = vpa(b(4))         % wyznaczanego z zadana precyzja
widmo_amp =abs(a)      % Widmo amplitudowe
% Widmo fazowe uporzadkowane monotonicznie
widmo_faz =unwrap(angle(a))
% 3 SPOSOB. Z formuly definicyjnej transformacji Fouriera
s=zeros(1,N);
for k=0:N-1
    for n=0:N-1
        s(k+1)=s(k+1)+x(n+1)*(exp(-j*2*pi*n*k/N)));
    end
end
c4 = vpa(s(4))     % (vpa: Matlab variable-precision arithmetic)
```

Po wykonaniu powyższego programu można odczytać wartość p i ustalić, że $p \neq 0$, co oznacza, że w zależności od przyjętej metody wyznaczania widma oraz założonej dokładności obliczeń wartości poszczególnych współczynników widmowych mogą się różnić. Różnice wyników, choć mało znaczące, wynikają ze sposobów przeprowadzania wewnętrznych obliczeń w Matlabie. Na przykład wartości czwartego współczynnika widmowego wyznaczonego w poprzednim przykładzie różnymi sposobami są następujące:

$$
\begin{aligned}
a(4) &= -8.0000000000000000000 + 11.972846101323911583j,\\
b(4) &= -8.0000000000000000000 + 11.972846101323913359j,\\
c(4) &= -8.0000000000000621725 + 11.972846101323973755j.
\end{aligned}
\tag{5.30}
$$

W większości zastosowań aplikacyjnych nieistotne różnice wyników mogą być ignorowane. Większe znaczenie ma np. złożoność czasowa i obliczeniowa algorytmu, gdzie dla długich ciągów danych wejściowych są to parametry krytyczne.

Program 5.2. *Stosując oznaczenia zmiennych jak w przykładzie poprzednim, można odtworzyć pierwotne wartości próbek, czyli wyznaczyć wartości elementów wektora* $\boldsymbol{x}$.

```
y3 = ifft(a);                    % Za pomoc predefiniowanej
                                 % funkcji ifft()
y4 = b*conj(dftmtx(N))/N;        % To samo, ale za pomoca funkcji
                                 % dftmtx()
x=zeros(1,N);                    % Rezerwacja miejsca na wynik
for k=0:N-1
   for n=0:N-1                   % Odtworzenie danych
      x(n+1)=x(n+1)+s(k+1)*(exp(j*2*pi*n*k/N)));
   end
end
```

5.2.2. Szybka dyskretna transformacja Fouriera

Po spełnieniu pewnych warunków dyskretną transformatę Fouriera (DFT) można wyznaczyć za pomocą algorytmu Cooleya–Tukeya, którego złożoność obliczeniowa (liczona jako liczba dominujących operacji) jest mniejsza w porównaniu z metodą bezpośrednią, w której stosujemy mnożenie macierzowe (5.28). Idea tego algorytmu polega na odpowiedniej organizacji obliczeń. Zamiast klasycznej N punktowej transformacji DFT wyznaczane są dwie $N/2$ punktowe transformacje, a wyniki są następnie łączone. Daje to w efekcie czasową złożoność obliczeniową $O(2 \cdot (\frac{N}{2})^2) = O(\frac{N^2}{2})$.

W szacowaniu złożoności nie bierze się pod uwagę czasu potrzebnego na łączenie widm, gdyż czas wykonania operacji łączenia jest bardzo krótki. Podejście takie jest bardzo opłacalne. Algorytm tego typu oznaczany jest anglojęzycznym akronimem FFT (ang. *Fast Fourier Transform*). Jego wersja odwrotna, na podstawie której odtwarzamy oryginalne wartości wektora próbek, nosi nazwę IFFT (ang. *Inverse Fast Fourier Transform*). Prosta oraz odwrotna szybka transformacja Fouriera są obecnie standardowymi, predefiniowanymi bibliotecznymi funkcjami większości pakietów matematycznych i elementem bibliotek wielu języków programowania ogólnego przeznaczenia.

Klasyfikacja algorytmów FFT prowadzi do ich podziałów na różne typy. Najważniejszym kryterium klasyfikacji jest długość wejściowego wektora danych N. Wyróżnia się algorytmy Cooleya–Tukeya i tzw. algorytmy Prime Factor. Do pierwszej grupy zaliczane są algorytmy, w których rozmiar N wektora danych jest całkowitą potęgą liczby 2. W drugiej grupie znajdują się algorytmy, w których wymieniony warunek na długość wektora danych nie jest spełniony – można wtedy długość wektora danych przedstawić jako iloczyn liczb pierwszych.

Klasyfikację można również przeprowadzić ze względu na podstawę podziału (ang. *radix*). Otrzymuje się wtedy algorytmy typu radix–k, $k = 2, 4, 8$, itd. Algorytmy można również klasyfikować ze względu na dziedzinę podziału. Wyróżnia się algorytmy z podziałem w czasie DIT (ang. *Decimation In Time*) oraz z podziałem w częstotliwości DIF (ang. *Decimation In Frequency*). Najpopularniejszą wersją algorytmu Cooleya–Tukeya jest algorytm Radix-2 DIT. Zaprogramowanie algorytmu w wersji szybkiej wymaga, aby liczba próbek sygnału pierwotnego była potęgą liczby 2 [11,40]. Cooley i Tukey zauważyli, że jeśli odpowiednio przestawić kolejność próbek wektora wejściowego, to złożoność obliczeniową dyskretnej transformacji Fouriera można zredukować z $O(N^2)$ do $O(N \log_2 N)$. Liczba iteracji w tym algorytmie wynosi $\log_2 N$, a w każdej iteracji wykonuje się $N/2 \log_2 N$ zespolonych mnożeń oraz $N \log_2 N$ dodawań. Aby lepiej zauważyć te związki, równanie transformacji Fouriera (5.16) należy zapisać w postaci:

$$s(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x(n) w_N^{kn}, \quad k = 0, 1, ..., N-1, \quad w_N = e^{-j\frac{2\pi}{N}}. \tag{5.31}$$

Wielkość pomocnicza $w_N = e^{-j\frac{2\pi}{N}}$ jest równa pierwiastkowi N-tego stopnia z jedności. Wielkość w jest liczbą zespoloną o jednostkowym module i argumencie wyrażonym ilorazem $2\pi/N$. Wielkość ta odgrywa w teorii sygnałów dyskretnych ważną rolę, gdyż pozwala zespolone sygnały harmoniczne $e^{-j2\pi\frac{n}{N}k}$ występujące we wzorze (5.31) zapisywać w formie skróconej:

$$e^{-j2\pi\frac{n}{N}k} = (e^{-j\frac{2\pi}{N}})^{kn} = w_N^{kn}. \tag{5.32}$$

Funkcje wykładnicze $w_N^{kn} = e^{-j\frac{2\pi}{N}kn} = \cos(\frac{2\pi}{N}kn) - j\sin(\frac{2\pi}{N}kn)$ są więc wektorami jednostkowymi w płaszczyźnie zespolonej z kosinusoidalną częścią rzeczywistą i sinusoidalną częścią urojoną. Indeks N wektora w jest związany z liczbą elementów wektora, a wykładnik określa położenie wektora w płaszczyźnie zespolonej. Są to funkcje N okresowe:

$$w_N^{(k+N)n} = w_N^{kn} w_N^{Nn} = w_N^{kn} e^{-j2\pi n} = w_N^{kn}, \tag{5.33}$$

gdyż $e^{-j2\pi n} = \cos(2\pi n) - j\sin(2\pi n) = 1$ dla każdego $n = 0, \pm 1, \pm 2, ...$

Funkcje wykładnicze w_N^{kn} są parami ortogonalne, ponieważ:

$$\sum_{n=0}^{N-1} w_N^{kn} w_N^{-ln} = \begin{cases} N & \text{dla} \quad (k-l) = 0 \\ 0 & \text{w przeciwnym przypadku} \end{cases}. \tag{5.34}$$

Dla parzystego N próbki sygnału można tak porządkować, aby podzielić je na parzyste i nieparzyste:

$$s(k) = \sum_{n=0}^{N/2-1} x(2n) w_N^{k(2n)} + \sum_{n=0}^{N/2-1} x(2n+1) w_N^{k(2n+1)}, \quad k = 0, 1, ..., N-1. \tag{5.35}$$

Funkcje w_N^{kn} są symetryczne i mają następujące własności:

$$w_N^{kn} = w_N^{(k+N)n} = w_N^{k(n+N)},$$

$$w_N^{2kn} = e^{-j\frac{2\pi}{N}\cdot 2kn} = e^{-j\frac{2\pi}{N/2}\cdot kn} = w_{N/2}^{kn},$$

$$w_N^{N/2} = -1, \quad w_N^N = 1,$$

$$w_N^{(k+N/2)} = -w_N^k, \tag{5.36}$$

$$w_N^{(k+N)} = w_N^k.$$

Na podstawie (5.33) mamy:

$$w_N^{k(2n+1)} = w_N^{2kn} w_N^k = w_N^{kn} w_N^k. \tag{5.37}$$

Wzór (5.31) można więc zapisać ponownie w następującej formie:

$$s(k) = \sum_{n=0}^{N/2-1} x(2n) w_{N/2}^{kn} + w_N^k \cdot \sum_{n=0}^{N/2-1} x(2n+1) w_{N/2}^{kn}, \qquad k = 0, 1, ..., N-1. \tag{5.38}$$

W dwóch sumach występujących we wzorze (5.38) można rozpoznać $N/2$ punktowe równania dyskretnej transformacji Fouriera dla parzystych i nieparzystych próbek sygnału wejściowego $\boldsymbol{x}$. Daje to w efekcie pełną, N punktową transformację.

Uwzględniając w dalszych rozważaniach zależności (5.32)–(5.38), elementy w_N^{kn} macierzy $\boldsymbol{W}_N$ można inaczej uporządkować, formując tym samym nową macierz $\boldsymbol{W}_N^{\%}$ składającą się z kolejno ułożonych najpierw parzystych, a potem nieparzystych kolumn macierzy $\boldsymbol{W}_N$. Nowy porządek pozwoli w efekcie na szybsze wykonywanie obliczeń widma i ułatwi organizację obliczeń numerycznych zarówno w wersji programowej, jak i sprzętowej. Takie rozwiązania są dzisiaj powszechnie znane.

$$\boldsymbol{W}_N = \begin{bmatrix} w_N^{0\cdot 0} & w_N^{0\cdot 1} & w_N^{0\cdot 2} & \cdots & w_N^{0\cdot(N-2)} & w_N^{0\cdot(N-1)} \\ w_N^{1\cdot 0} & w_N^{1\cdot 1} & w_N^{1\cdot 2} & \cdots & w_N^{1\cdot(N-2)} & w_N^{1\cdot(N-1)} \\ w_N^{2\cdot 0} & w_N^{2\cdot 1} & w_N^{2\cdot 2} & \cdots & w_N^{2\cdot(N-2)} & w_N^{2\cdot(N-1)} \\ \vdots & & \cdots & & & \vdots \\ w_N^{(N-1)\cdot 0} & w_N^{(N-1)\cdot 1} & w_N^{(N-1)\cdot 2} & \cdots & w_N^{(N-1)\cdot(N-2)} & w_N^{(N-1)\cdot(N-1)} \end{bmatrix}, \tag{5.39}$$

$$\boldsymbol{W}_N^{\%} = \begin{bmatrix} w_N^{0\cdot 0} & w_N^{0\cdot 2} & \cdots & w_N^{0\cdot(N-2)} & w_N^{0\cdot 1} & \cdots & w_N^{0\cdot(N-1)} \\ w_N^{1\cdot 0} & w_N^{1\cdot 2} & \cdots & w_N^{1\cdot(N-2)} & w_N^{1\cdot 1} & \cdots & w_N^{1\cdot(N-1)} \\ w_N^{2\cdot 0} & w_N^{2\cdot 2} & \cdots & w_N^{2\cdot(N-2)} & w_N^{2\cdot 1} & \cdots & w_N^{2\cdot(N-1)} \\ \vdots & & \cdots & & & & \vdots \\ w_N^{(N-1)\cdot 0} & w_N^{(N-1)\cdot 2} & \cdots & w_N^{(N-1)\cdot(N-2)} & w_N^{(N-1)\cdot 1} & \cdots & w_N^{(N-1)\cdot(N-1)} \end{bmatrix}. \tag{5.40}$$

Kwadratowa macierz (5.40) ma taką budowę, że można ją również przedstawić w postaci macierzy blokowej składającej się z podmacierzy o dwa razy mniejszych rozmiarach:

$$\boldsymbol{W}_N^{\%} = \begin{bmatrix} \boldsymbol{V}_{N/2} & \boldsymbol{G}_{N/2} \\ \boldsymbol{V}_{N/2} & -\boldsymbol{G}_{N/2} \end{bmatrix}. \tag{5.41}$$

Wartości elementów macierzy $\boldsymbol{G}_{N/2}$ można jednak wyznaczyć bezpośrednio z elementów macierzy $\boldsymbol{V}_{N/2}$ na podstawie formuły:

$$[o_{kn}]_{\frac{N}{2}\times\frac{N}{2}} = e^{-j\frac{2\pi}{N}\cdot k}[v_{kn}]_{\frac{N}{2}\times\frac{N}{2}}, \quad k,n = 0,...,\frac{N}{2}-1, \tag{5.42}$$

gdzie k, n są odpowiednimi numerami wierszy i kolumn macierzy $\boldsymbol{V}_{N/2}$ i $\boldsymbol{G}_{N/2}$. Tak więc:

$$\boldsymbol{W}_N^{\%} = \begin{bmatrix} \boldsymbol{V}_{N/2} & +\boldsymbol{O}_{N/2}\boldsymbol{V}_{N/2} \\ \boldsymbol{V}_{N/2} & -\boldsymbol{O}_{N/2}\boldsymbol{V}_{N/2} \end{bmatrix}, \tag{5.43}$$

gdzie:

$$\boldsymbol{O}_{N/2} = \begin{bmatrix} e^{-j\frac{2\pi}{N}\cdot 0} & 0 & \cdots & 0 \\ 0 & e^{-j\frac{2\pi}{N}\cdot 1} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & e^{-j\frac{2\pi}{N}\cdot(\frac{N}{2}-1)} \end{bmatrix}_{\frac{N}{2}\times\frac{N}{2}} . \tag{5.44}$$

Do budowy macierzy $\boldsymbol{W}_N^{\%}$ wystarczy więc tylko znajomość macierzy $\boldsymbol{V}_{N/2}$. Łatwo zauważyć, że macierze $\boldsymbol{V}$ formują pierwszy składnik sumy we wzorze (5.38), a macierze $\boldsymbol{G}$ – składnik drugi.

Proces dekompozycji (faktoryzacji) macierzy $\boldsymbol{W}_N^{\%}$ można kontynuować dla macierzy $\boldsymbol{W}_{N/2}^{\%}, \boldsymbol{W}_{N/4}^{\%}, \ldots$.

W celu faktoryzacji każdej nowej dwa razy mniejszej macierzy należy ponownie ją uporządkować (najpierw parzyste, a potem nieparzyste kolumny). Proces podziału macierzy $\boldsymbol{W}_N^{\%}$ zostaje zakończony dla $W_2^{\%}$:

$$\boldsymbol{W}_4^{\%} = \begin{bmatrix} \boldsymbol{V}_2 & \boldsymbol{G}_2 \\ \boldsymbol{V}_2 & -\boldsymbol{G}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_2 & +\boldsymbol{O}_2\boldsymbol{V}_2 \\ \boldsymbol{V}_2 & -\boldsymbol{O}_2\boldsymbol{V}_2 \end{bmatrix}, \tag{5.45}$$

gdzie:

$$\begin{aligned} \boldsymbol{V}_2 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \boldsymbol{G}_2 = \begin{bmatrix} 1 & 1 \\ -j & j \end{bmatrix} \\ \boldsymbol{O}_2 &= \begin{bmatrix} e^{-j\frac{2\pi}{4}\cdot 0} & 0 \\ 0 & e^{-j\frac{2\pi}{4}\cdot 1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -j \end{bmatrix} \end{aligned} , \tag{5.46}$$

$$\boldsymbol{W}_2^{\%} = \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{G}_1 \\ \boldsymbol{V}_1 & -\boldsymbol{G}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_1 & +\boldsymbol{O}_1\boldsymbol{V}_1 \\ \boldsymbol{V}_2 & -\boldsymbol{O}_1\boldsymbol{V}_1 \end{bmatrix}, \tag{5.47}$$

gdzie:

$$\begin{aligned} \boldsymbol{V}_1 &= [1], \qquad \boldsymbol{G}_2 = [1] \\ \boldsymbol{O}_1 &= \left[e^{-j\frac{2\pi}{2}\cdot 0}\right] = 1 \end{aligned} . \tag{5.48}$$

Proces dekompozycji można także odwrócić – z macierzy elementarnej $\boldsymbol{W}_2^{\%}$ można ponownie zbudować macierz oryginalną $\boldsymbol{W}_N^{\%}$. Przedstawiony powyżej proces przekształcania macierzy $\boldsymbol{W}_N^{\%}$ wraz ze zmianą uporządkowania próbek danych stanowi fundament szybkiej transformacji Fouriera. Zaprezentowaną ideę dekompozycji macierzy $\boldsymbol{W}_N^{\%}$ można wykorzystać do budowy algorytmu szybkiej transformacji Fouriera – FFT. Dzięki temu algorytmowi pojawiły się możliwości praktycznego przetwarzania sygnałów cyfrowych, a także stosowania szybkich dyskretnych transformat kosinusowych, sinusowych i wielu innych.

Przykład 5.5. *Metodą szybkiej transformacji Fouriera wyznaczyć widmo wektora próbek* $\boldsymbol{x} = [x_0, x_1, x_3, x_4, x_5, x_6, x_7]$. *Na podstawie* (5.24) *otrzymujemy rozwiązanie klasyczne*:

$$\boldsymbol{s} = \boldsymbol{W}_8 \cdot \boldsymbol{x} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1-j}{\sqrt{2}} & -j & \frac{-1-j}{\sqrt{2}} & -1 & \frac{-1+j}{\sqrt{2}} & j & \frac{1+j}{\sqrt{2}} \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & \frac{-1-j}{\sqrt{2}} & j & \frac{1-j}{\sqrt{2}} & -1 & \frac{1+j}{\sqrt{2}} & -j & \frac{-1+j}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{-1+j}{\sqrt{2}} & -j & \frac{1+j}{\sqrt{2}} & -1 & \frac{1-j}{\sqrt{2}} & j & \frac{-1-j}{\sqrt{2}} \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & \frac{1+j}{\sqrt{2}} & j & \frac{-1+j}{\sqrt{2}} & -1 & \frac{-1-j}{\sqrt{2}} & -j & \frac{1-j}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}.$$

Permutując kolumny macierzy $\boldsymbol{W}_8$ *oraz elementy wektora danych, otrzymujemy*:

$$\boldsymbol{s} = \boldsymbol{W}_8^{\%} \cdot \boldsymbol{x}^{\%} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j & \frac{1-j}{\sqrt{2}} & \frac{-1-j}{\sqrt{2}} & \frac{-1+j}{\sqrt{2}} & \frac{1+j}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & -j & j & -j & j \\ 1 & j & -1 & -j & \frac{-1-j}{\sqrt{2}} & \frac{1-j}{\sqrt{2}} & \frac{1+j}{\sqrt{2}} & \frac{-1+j}{\sqrt{2}} \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -j & -1 & j & \frac{-1+j}{\sqrt{2}} & \frac{1+j}{\sqrt{2}} & \frac{1-j}{\sqrt{2}} & \frac{-1-j}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & j & -j & j & -j \\ 1 & j & -1 & -j & \frac{1+j}{\sqrt{2}} & \frac{-1+j}{\sqrt{2}} & \frac{-1-j}{\sqrt{2}} & \frac{1-j}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \\ x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix}.$$

Macierz $\boldsymbol{W}_8^{\%}$ *można poddać faktoryzacji*:

$$\boldsymbol{s} = \boldsymbol{W}_8^{\%} \cdot \boldsymbol{x}^{\%} = \begin{bmatrix} & \\ \boldsymbol{V}_4 & +\boldsymbol{O}_4\boldsymbol{V}_4 \\ & \\ \boldsymbol{V}_4 & -\boldsymbol{O}_4\boldsymbol{V}_4 \\ & \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \\ x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix},$$

$$\boldsymbol{V}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}, \quad \boldsymbol{O}_4 = diag(e^{-j\frac{2\pi}{8}\cdot 0}, e^{-j\frac{2\pi}{8}\cdot 1}, e^{-j\frac{2\pi}{8}\cdot 2}, e^{-j\frac{2\pi}{8}\cdot 3}).$$

Powyższą formułę wyznaczania widma można przedstawić inaczej – w sposób równoważny. Polega on na odpowiednim podziale elementów wejściowego wektora danych $\boldsymbol{x}$:

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{V}_4 \cdot \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} & +\boldsymbol{O}_4\boldsymbol{V}_4 \cdot \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix} \\ \\ \boldsymbol{V}_4 \cdot \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} & -\boldsymbol{O}_4\boldsymbol{V}_4 \cdot \begin{bmatrix} x_1 \\ x_3 \\ x5 \\ x7 \end{bmatrix} \end{bmatrix},$$

gdzie:

$$\boldsymbol{V}_4 \cdot \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_2 & +\boldsymbol{O}_2\boldsymbol{V}_2 \\ \\ \boldsymbol{V}_2 & -\boldsymbol{O}_2\boldsymbol{V}_2 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} = \begin{bmatrix} \boldsymbol{V}_2 \cdot \begin{bmatrix} x_0 \\ x_4 \end{bmatrix} + \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_2 \\ x_6 \end{bmatrix} \\ \\ \boldsymbol{V}_2 \cdot \begin{bmatrix} x_0 \\ x_4 \end{bmatrix} - \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_2 \\ x_6 \end{bmatrix} \end{bmatrix}.$$

W omawianym przykładzie komplet współczynników widmowych można więc wyznaczyć następująco:

$$\boldsymbol{s} = \begin{bmatrix} \begin{bmatrix} \boldsymbol{V}_2 \cdot \begin{bmatrix} x_0 \\ x_4 \end{bmatrix} + \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_2 \\ x_6 \end{bmatrix} \\ \\ \boldsymbol{V}_2 \cdot \begin{bmatrix} x_0 \\ x_4 \end{bmatrix} - \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_2 \\ x_6 \end{bmatrix} \end{bmatrix} & +\boldsymbol{O}_4 \cdot \begin{bmatrix} \boldsymbol{V}_2 \cdot \begin{bmatrix} x_1 \\ x_5 \end{bmatrix} + \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_3 \\ x_7 \end{bmatrix} \\ \\ \boldsymbol{V}_2 \cdot \begin{bmatrix} x_1 \\ x_5 \end{bmatrix} - \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_3 \\ x_7 \end{bmatrix} \end{bmatrix} \\ \\ \begin{bmatrix} \boldsymbol{V}_2 \cdot \begin{bmatrix} x_0 \\ x_4 \end{bmatrix} + \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_2 \\ x_6 \end{bmatrix} \\ \\ \boldsymbol{V}_2 \cdot \begin{bmatrix} x_0 \\ x_4 \end{bmatrix} - \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_2 \\ x_6 \end{bmatrix} \end{bmatrix} & -\boldsymbol{O}_4 \cdot \begin{bmatrix} \boldsymbol{V}_2 \cdot \begin{bmatrix} x_1 \\ x_5 \end{bmatrix} + \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_3 \\ x_7 \end{bmatrix} \\ \\ \boldsymbol{V}_2 \cdot \begin{bmatrix} x_1 \\ x_5 \end{bmatrix} - \boldsymbol{O}_2\boldsymbol{V}_2 \cdot \begin{bmatrix} x_3 \\ x_7 \end{bmatrix} \end{bmatrix} \end{bmatrix}.$$

Dwuwymiarowe wektory wyznacza się tylko jednokrotnie, co znacznie skraca czas wyznaczania widma $\boldsymbol{s}$. Porównując elementy wektora próbek $\boldsymbol{x}$ w równaniu (5.5) z elementami dwuwymiarowych wektorów w równaniu (5.5), można zauważyć, że elementy wektora $\boldsymbol{x}$ poddane zostały permutacji:

$$\boldsymbol{x} = [x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7]^T \rightarrow \boldsymbol{x}^r = [x_0, x_4, x_2, x_6, x_1, x_5, x_3, x_7]^T.$$

Jeśli każdemu wyrażonemu dziesiętnie numerowi indeksu i elementu $x_i \in \boldsymbol{x}$ oraz $x_i \in \boldsymbol{x}^r$ przypisać jego binarny odpowiednik, to okaże się, że elementy wektora $\boldsymbol{x}$ zostały uporządkowane w odwrotnej kolejności bitowej (ang. *bit reversal order*) indeksów, tworząc wektor $\boldsymbol{x}^r$.

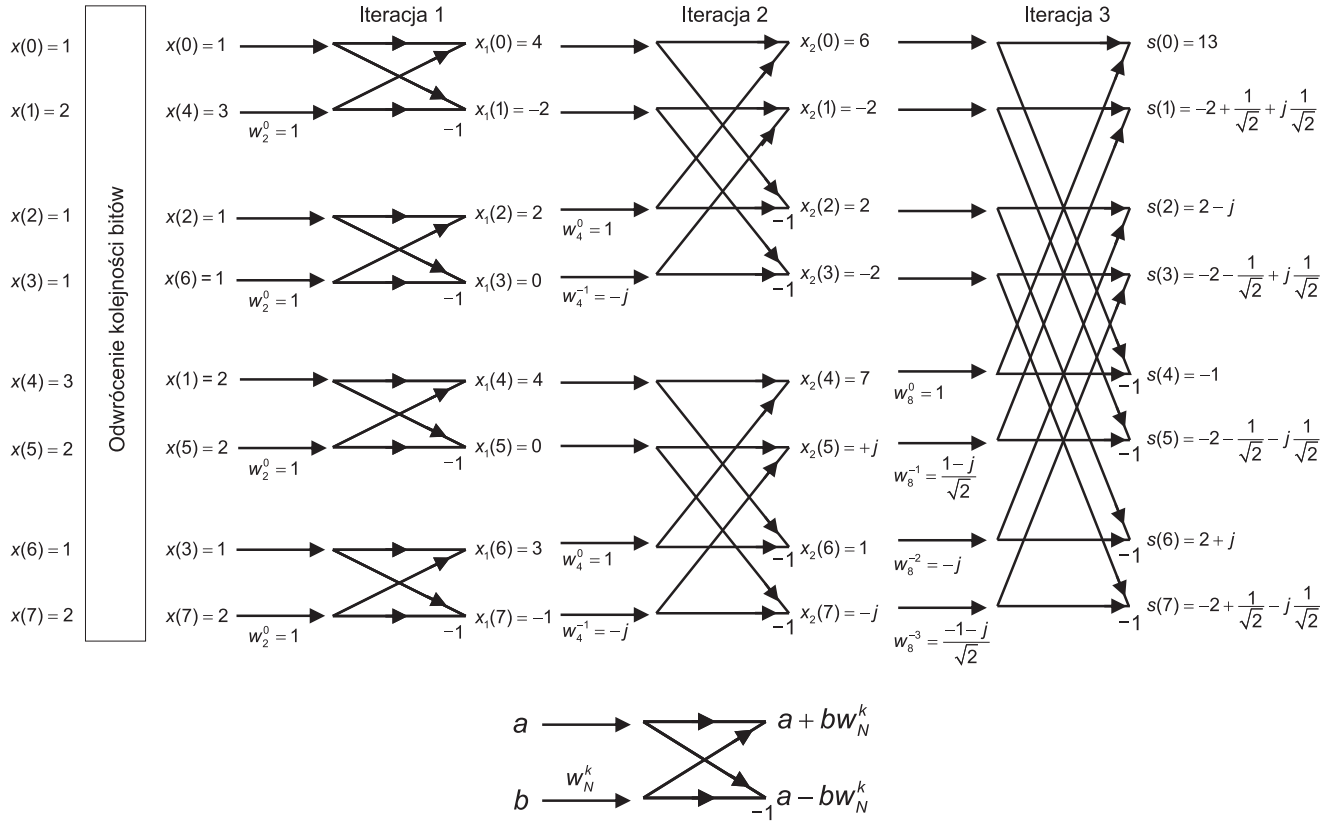
Z równania (5.5) wynika, że wszystkie obliczenia dla dowolnej wartości N można sprowadzić do operacji mnożenia dwuwymiarowych wektorów przez macierz $\boldsymbol{V}_2$ lub macierz $\boldsymbol{O}_2\boldsymbol{V}_2$. Ponieważ wyznaczanie macierzy odbywa się jednorazowo, proces obliczeń jest szybszy. Przedstawione w tym rozdziale wywody oraz detale Przykładu 5.5 pozwalają na zaprezentowanie uniwersalnego schematu obliczeń Radix-2 szybkiej transformacji Fouriera dla dowolnych $N = 2^p$, gdzie p jest liczbą naturalną.

Organizacja prostej, szybkiej transformacji Fouriera przedstawiona została w postaci schematu motylkowego na rys. 3. W celach poglądowych pokazano również wyniki obliczeń uzyskiwane w kolejnych krokach iteracji. W tego typu algorytmie dane mogą być nadpisywane, gdyż informacje wejściowe każdej operacji motylkowej wykorzystywane są tylko raz. Jeśli dane na wejściu pojedynczego motylka oznaczymy przez a oraz b, to pojedyncza operacja motylkowa może być opisana następującym pseudokodem:

$$\begin{aligned} zm &\leftarrow b \cdot w_N^k \\ a &\leftarrow a + zm \\ b &\leftarrow a - zm \end{aligned} \quad .$$

Schemat motylkowy zaprezentowany na rys. 3 wskazuje, że w każdej operacji motylkowej wykonywane są dwa dodawania oraz jedno mnożenie. Oznacza to, że wyznaczenie kompletu współczynników widmowych wymaga wykonania $N \log_2 N$ zespolonych dodawań/odejmowań oraz $N/2 \log_2 N$ zespolonych mnożeń, co w notacji O daje złożoność rzędu $O(N \log_2 N)$.

Należy pamiętać o dużych korzyściach, jakie przyniosły rozwiązania proponowane przez Cooleya–Tukeya. Do lat 60. ubiegłego wieku obliczenie dyskretnej transformaty Fouriera, nawet dla niewielkich N, było uciążliwe, a dla dużych N wręcz niemożliwe. Algorytm Cooleya–Tukeya jest najbardziej rozpowszechnionym algorytmem szybkiej transformacji Fouriera, stosowanym do dzisiaj w licznych odmianach. Jako ciekawostkę można wskazać, że już w roku 1805 zadanie to rozwiązał Gauss, ale jego praca, napisana po łacinie, nie była powszechnie znana.



Rys. 3. Schemat algorytmu szybkiej transformacji Fouriera Radix-2 DIT dla $N = 8$ wraz z przykładem obliczeń cząstkowych w poszczególnych krokach iteracji

Transformacja odwrotna – IFFT (ang. *Inverse Fast Fourier Transform*), czyli odtworzenie wektora danych na podstawie widma, przebiega podobnie. Transformacje (5.16) oraz (5.17) różnią się tylko znakiem wykładnika funkcji wykładniczej. Schemat motylkowy z rys. 3 można więc łatwo adaptować do obliczeń transformacji odwrotnej – wystarczy zamiast mnożników w_N^{kn} stosować mnożniki w_N^{-kn}. Organizacja schematu motylkowego transformacji odwrotnej nie wymaga więc żadnych zmian. Poniżej przedstawiono program realizujący FFT lub IFFT (zgodny ze schematem prezentowanym na rys. 3).

Program 5.3. *Program prostej i odwrotnej szybkiej, jednowymiarowej transformacji Fouriera. Wersja szybka wymaga jednak, aby liczba elementów N wektora danych była potęgą dwójki oraz $N \geq 2$.*

```
% Szybka transformacja Fouriera. Wersja wg schematu motylkowego
rodzaj=1                          % 1 lub 2: transformacja
                                  % prosta lub odwrotna
A=[1 2 3 j];                    % Wektor polegajacy transformacji
Fast_F(A,1)                     % Funkcja Fast_F
%  CIALO FUNKCJI - dla obliczenia widma Fouriera
function X=Fast_F(dane,rodzaj)
N=pow2(floor(log2(length(dane)))); % Liczba N musi byc potega 2!!
[f,e]=log2(length(dane));
I=dec2bin(0:pow2(0.5,e)-1);       % Binarna numeracja elementow
                                  % wektora
R=dane(bin2dec(I(:,e-1:-1:1))+1); % Odwrocenie kolejnosci bitow
dane=R;                           % Nowy porzadek
X=dane(1:N);                      % Wektor danych
p1=2; p2=N/2; p3=1;               % w nowym porzadku
for zm1=1:log2(N)
    d1=1;
    if (rodzaj==1)                % Rodzaj transformacji
       W=exp(-j*2*pi/2^zm1);
    else
```

```
       W=exp(j*2*pi/2^zm1);
    end
    for zm2=1:p2
        for zm3=1:p3
            i=zm3+d1-1; v=i+p3;
            A=X(i); B=X(v)*W^(zm3-1);
            X(i)=A+B;          % Elementarne operacje motylkowe
            X(v)=A-B;          % Elementarne operacje motylkowe
        end
        d1=d1+p1;
    end
    p1=p1*2; p2=p2/2;  p3=p3*2;;
end
dane=X;
if(rodzaj==2)
  dane=(1/N)*X                % Transformacja odwrotna
end
```

Należy również podkreślić, że techniki obliczeń widma wpływają nie tylko na dokładność, ale także na czas obliczeń. W środowisku Matlab wywołanie funkcji **fft**($\boldsymbol{x}$) zapewnia wykonanie obliczeń w tzw. wersji szybkiej, gdzie liczba elementarnych operacji mnożenia i dodawania jest najmniejsza, a $\boldsymbol{x}$ jest wektorem danych, dla których wyznaczamy transformatę. Jest wiele, mających konkretne zastosowania, algorytmów wyznaczania widma Fouriera. Ich budowa zależy od rozwiązywanego problemu oraz od charakterystycznych własności sprzętu, na którym te algorytmy są implementowane. Dostępne są również specjalizowane układy scalone autonomicznie, realizujące szybką transformację Fouriera. Zagadnienia te zostały przejrzyście, z dbałością o detale, omówione w pracach [11, 17, 40]. Jednowymiarową transformatę Fouriera można rozszerzyć na transformaty wielowymiarowe. W wielu zastosowaniach praktycznych (np. w przetwarzaniu obrazów) mamy do czynienia z transformacjami dwuwymiarowymi.

Czasem trzeba przeprowadzić transformację dla ciągów o dowolnej liczbie elementów, których liczba niekoniecznie jest potęgą liczby 2. Obliczenia można wtedy wykonać na podstawie wzorów definicyjnych (5.16) oraz (5.17).

Program 5.4. *Program prostej oraz odwrotnej jednowymiarowej transformacji Fouriera dla dowolnej długości ciągu danych.*

```
A=[1 2 3 4 j];           % Wektor danych o dowolnej dlugosci
N=length(A);             % Wyznaczenie dlugosci wektora A
rodzaj=1;                % 1 lub 2: transformacja prosta
                         % lub odwrotna
F=zeros(N);              % Deklaracja macierzy Fouriera
for k=1:N
    for w=1:N
        if(rodzaj==1)      % Fourierowska macierz transformacji
          F(w,k)=exp(-j*2*pi*(w-1)*(k-1)/N);
        else
          F(w,k)=exp(j*2*pi*(w-1)*(k-1)/N);
       end
    end
end
if(rodzaj==1)
   widmo=A*F             % Transformata wektora A
else
   widmo=1/N*A*F
end
```

Obliczenia przeprowadzone według zasad podanych w Przykładzie 5.4 będą zgodne z wynikami uzyskanymi za pomocą wywołania bibliotecznej funkcji Matlaba **fft**() lub **ifft**().

5.2.3. Dwuwymiarowa transformacja Fouriera

Definicja 5.5. *Dwuwymiarową dyskretną transformacją Fouriera (ang. 2D Discrete Fourier Transform – 2D-DFT) ciągu danych* $x(m,n)$, $m = 0,1,...,M-1$, $n = 0,1,...,N-1$ *nazywamy ciąg zespolony współczynników rozwinięcia*:

$$\begin{aligned} s(k,l) &= \alpha \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) \cdot e^{-j2\pi(\frac{m}{M}k+\frac{n}{N}l)}, \\ k &= 0,1,...,M-1, \quad l = 0,1,...,N-1. \end{aligned} \tag{5.49}$$

Według podobnych zasad można wyznaczyć odwrotną dwuwymiarową transfomację Fouriera.

Definicja 5.6. *Dwuwymiarową odwrotną transformacją Fouriera (ang. 2D Inverse Discrete Fourier Transform – 2D-IDFT) jest odwzorowanie, które na podstawie widma* $s(k,l)$, $k,0,...,M-1$, $l = 0,1,...,N-1$ *odtwarza sygnał pierwotny* $x(m,n)$:

$$\begin{aligned} x(m,n) &= \beta \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} s(k,l) \cdot e^{j2\pi(\frac{m}{M}k+\frac{n}{N}l)}, \\ m &= 0,1,...,M-1, \quad n = 0,1,...,N-1. \end{aligned} \tag{5.50}$$

Elementy $s(k,l)$ tworzą macierz o wymiarach $M \times N$, której kolumny są jednowymiarowymi dyskretnymi transformatami Fouriera. Operacje (5.49) oraz (5.50) są wzajemnie odwrotne, jeśli respektowany będzie warunek:

$$\alpha \cdot \beta = \frac{1}{M \cdot N}. \tag{5.51}$$

Występujący w przekształceniu (5.49) czynnik wykładniczy można zapisać w następującej postaci:

$$e^{-j2\pi(\frac{m}{M}k+\frac{n}{N}l)} = e^{-j2\pi\frac{m}{M}k} \cdot e^{-j2\pi\frac{n}{N}l}. \tag{5.52}$$

Podobny związek można wskazać dla przekształcenia (5.50). Przywołując omawianą wcześniej jednowymiarową transformację Fouriera, można zauważyć, że

fourierowską transformatę dwuwymiarową można przedstawić jako złożenie dwóch transformat jednowymiarowych:

$$s(k,l) = \alpha \sum_{n=0}^{N-1} \left[\sum_{m=0}^{M-1} x(m,n) \cdot e^{-j2\pi \frac{m}{M} k} \right] e^{-j2\pi \frac{n}{N} l}, \tag{5.53}$$

$$x(m,n) = \beta \sum_{l=0}^{N-1} \left[\sum_{k=0}^{M-1} s(k,l) \cdot e^{j2\pi \frac{m}{M} k} \right] e^{j2\pi \frac{n}{N} l}. \tag{5.54}$$

Dwuwymiarową fourierowską dyskretną prostą i odwrotną transformację można opisać macierzowo w sposób następujący:

$$\boldsymbol{S} = \alpha \cdot \boldsymbol{W}_a \cdot \boldsymbol{X} \cdot \boldsymbol{W}_b, \tag{5.55}$$

$$\boldsymbol{X} = \beta \cdot \overline{\boldsymbol{W}_a} \cdot \boldsymbol{S} \cdot \overline{\boldsymbol{W}_b}, \tag{5.56}$$

gdzie $\boldsymbol{S}, \boldsymbol{X} \in \boldsymbol{C}^{M \times N}$, $\boldsymbol{W}_a = [w_{m,k}]_{M \times M} = e^{-j\frac{2\pi}{M}mk}$,

$\boldsymbol{W}_b = [w_{n,l}]_{N \times N} = e^{-j\frac{2\pi}{N}nl}$, a $\overline{\boldsymbol{W}_a}$ oraz $\overline{\boldsymbol{W}_b}$ są odpowiednimi macierzami sprzężonymi z $\boldsymbol{W}_a$ i $\boldsymbol{W}_b$.

Jeśli dane można opisać macierzą kwadratową $\boldsymbol{W} = [w_{m,k}]_{N \times N} = e^{-j\frac{2\pi}{N}mk}$, to:

$$\boldsymbol{S} = \alpha \cdot \boldsymbol{W} \cdot \boldsymbol{X} \cdot \boldsymbol{W}, \tag{5.57}$$

$$\boldsymbol{X} = \beta \cdot \overline{\boldsymbol{W}} \cdot \boldsymbol{S} \cdot \overline{\boldsymbol{W}}. \tag{5.58}$$

Transformacje Fouriera, a w szczególności jej jedno- i dwuwymiarowe odmiany, są jednymi z najlepiej rozpoznanych i opisanych metod przetwarzania sygnałów – stąd bardzo wiele metod obliczania jej współczynników widmowych. Wynika to z różnych funkcjonalności języków programowania i możliwości oferowanych przez różne dedykowane pakiety matematyczne. Poniżej przedstawiono program w języku Matlab do wyznaczania transformaty dwuwymiarowego sygnału, reprezentowanego macierzą M o rozmiarach $M \times N$. W programie zaprezentowano różne sposoby wyznaczania prostej oraz odwrotnej transformaty, pokazując wybrane sposoby programowania tego zagadnienia.

Program 5.5. *Prosta i odwrotna dwuwymiarowa transformacja Fouriera. Rozwiązanie programowe dla macierzy danych $M \times N$.*

```
A=round(2*rand(5,4));       % A jest macierza o wymiarach M X N
[M,N]=size(A);              % Wymiary M oraz N macierzy A
widmo_1=fft(fft(A.').')     % Widmo ze zlozenia dwoch 1D fft()
oryginal_1=ifft(ifft(widmo_1.').')
widmo_2=dftmtx(M)*A*dftmtx(N)  % Funkcja dftmtx() z biblioteki
oryginal_2=1/(M*N)*conj(dftmtx(M))*widmo_2*conj(dftmtx(N))
widmo_3=fft2(A)             % Funkja fft2() z biblioteki
oryginal_3=ifft2(widmo_3)   % Odtworzenie sygnalu pierwotnego
F1=zeros(M);;
% Wyznaczane transformaty z wzorow definicyjnych
for k=1:M
    for w=1:M
        F1(w,k)=exp(-j*2*pi*(w-1)*(k-1)/M);
    end
end
F2=zeros(N);
for k=1:N
    for w=1:N
        F2(w,k)=exp(-j*2*pi*(w-1)*(k-1)/N);
    end
end
widmo_4=F1*A*F2               % Widmo sygnalu dwuwymiarowego
% Odtworzenie sygnalu pierwotnego
oryginal_4=1/(M*N)*((F1').')*widmo_4*((F2').')
```

Transformację dwuwymiarową można również zrealizować jako złożenie dwóch jednowymiarowych transformacji Fouriera. Transformacja jednowymiarowa została omówiona wcześniej w postaci odpowiedniego programu komputerowego.

Program 5.6. *Realizacja dwuwymiarowej transformacji Fouriera jako złożenie transformacji jednowymiarowych.*

```
% Szybka dwuwymiarowa transformacja Fouriera
rodzaj=1                        % 1 lub 2: prosta lub odwrotna
A=[1 2 3 j; 2 3 7*j 4];         % Macierz (lub wektor) danych
Fast2_F(A,1)                    % Funkcja Fast2_F
%  CIALO FUNKCJI - dla obliczenia widma Fouriera
function X=Fast2_F(dane,rodzaj)
[M N]=size(dane);
M=pow2(floor(log2(M)));         % Liczba M musi byc potega 2!!
N=pow2(floor(log2(N)));         % Liczba N musi byc potega 2!!
X=zeros(M,N);
if (rodzaj==1)
   for k=1:M
       X(k,:)=Fast_F(dane(k,:),1); % Funkcja z Programu 5.3
   end                              % Transformacja wierszami
   if (M~=1)
      for k=1:N
          X(:,k)=Fast_F(X(:,k),1);  % Transformacja kolumnami
      end
   end
else
     for k=1:M
         X(k,:)=Fast_F(dane(k,:),2);  % Funkcja z Programu 5.3
     end                        % Transformacja odwr. (wierszami)
     if (M~=1)
        for k=1:N
            X(:,k)=Fast_F(X(:,k),2); % Transformacja kolumnami
        end
     end
end
```

Przyśpieszanie obliczeń jest możliwe w przypadku stosowania transformacji szybkich, gdzie żąda się, aby spełniony był warunek $N = 2^p$, a p jest liczbą naturalną. W praktyce takie założenia są często akceptowane. W przetwarzaniu obrazów jesteśmy na przykład często zainteresowani oceną cech obrazu, gdzie podstawą oceny jest interpretacja widma obrazu. W środowisku Matlab dwuwymiarową transformację Fouriera zrealizować można za pomocą wbudowanej funkcji **fft2**($\boldsymbol{X}$), gdzie $\boldsymbol{X}$ jest macierzą, dla której wyznacza się macierz transformaty $\boldsymbol{Y}$. Transformację odwrotną można wykonać za pomocą funkcji **ifft2**($\boldsymbol{Y}$).

Złożoność dwuwymiarowej transformacji Fouriera można zredukować w stosunku do klasycznego rachunku macierzowego, gdzie złożoność obliczeniowa wynosi $O(N^4)$. Ponieważ transformacja Fouriera jest separowalna i można wykonywać ją oddzielnie dla wierszy i kolumn macierzy wejściowej, jej złożoność redukuje się do $O(2N^2 \log N)$.

5.3. Dyskretna transformacja kosinusowa

Transformacja kosinusowa jest dobrze udokumentowaną metodą transformacji danych, a jej popularność wynika przede wszystkim z zastosowań w stratnej kompresji obrazów. Dla określenia transformacji stosowany jest często anglojęzyczny akronim DCT (ang. *Discrete Cosine Transform*).

DCT to liniowa odwracalna funkcja $f : \boldsymbol{R}^N \to \boldsymbol{R}^N$ przekształcająca ciąg liczb $x(0), x(1), ..., x(N-1)$ w inny ciąg liczb $s(0), s(1), ..., s(N-1)$. Dyskretna transformata kosinusowa wykorzystuje rozwinięcie sygnału w bazie ortogonalnych wielomianów Czebyszewa [28]:

$$\left\{ \frac{1}{\sqrt{N}}, \sqrt{\frac{2}{N}} \cos \frac{\pi k(2n+1)}{2N} \right\}, \quad k, n = 1, 2, ..., N-1, \tag{5.59}$$

a macierz dyskretnego przekształcenia kosinusowego zbudowana jest ze zdyskretyzowanych wielomianów Czebyszewa. DCT występuje w wielu wariantach opisywanych w literaturze jako: DCT-I, DCT-II, DCT-III, DCT-IV [37,40].

5.3.1. Jednowymiarowa transformacja kosinusowa

Definicja 5.7. *Jednowymiarową dyskretną transformacją kosinusową (ang. Discrete Cosine Transform) rzeczywistych wartości sygnału $x(n)$, określonych dla $n = 0, 1, ..., N-1$, nazywamy ciąg rzeczywistych współczynników rozwinięcia sygnału $x(n)$:*

$$s(k) = c(k) \cdot \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi k(2n+1)}{2N}\right), \qquad k = 0, 1, ..., N-1, \tag{5.60}$$

gdzie: $c(0) = \sqrt{1/N}$, $c(k) = \sqrt{2/N}$ dla $k > 0$.

Definicja 5.8. *Jednowymiarową dyskretną odwrotną transformacją kosinusową (ang. Inverse Discrete Cosine Transform) jest odwzorowanie, które na podstawie widma sygnału $s(k)$, $k = 0, ..., N-1$, odtwarza próbki sygnału pierwotnego $x(n)$, $n = 0, ..., N-1$:*

$$x(n) = \sqrt{\frac{1}{N}} s(0) + \sqrt{\frac{2}{N}} \cdot \sum_{k=1}^{N-1} s(k) \cos\left(\frac{\pi k(2n+1)}{2N}\right), \quad n = 0, 1, ..., N-1. \tag{5.61}$$

Z definicji transformacji kosinusowej wynika, że jądro transformacji jest takie samo, jak jądro transformacji odwrotnej, co ułatwia organizację obliczeń numerycznych. Macierz jądrowa transformacji (5.60) ma postać:

$$\boldsymbol{W}_c = [w_{k,n}]_{N \times N} = c(k) \cos\left(\frac{\pi k(2n+1)}{2N}\right), \tag{5.62}$$

gdzie:

$$c(k) = \begin{cases} \sqrt{1/N} & \text{dla} \quad k = 0 \\ \sqrt{2/N} & \text{dla} \quad k > 0 \end{cases}. \tag{5.63}$$

W Matlabie macierz jądrową $\boldsymbol{W}_c$ otrzymujemy po wydaniu polecenia **dctmtx**(N). Transformacja kosinusowa jest transformacją liniową.

Tym samym można ją zapisać w formie macierzowej:

$$\boldsymbol{s} = \boldsymbol{x} \cdot \boldsymbol{W}_c. \tag{5.64}$$

Z ortogonalności macierzy $\boldsymbol{W}_c$ wynika, że:

$$\boldsymbol{W}_c^{-1} = \boldsymbol{W}_c^T \ , \tag{5.65}$$

co ułatwia przeprowadzenie przekształcenia odwrotnego, gdyż macierz odwrotna jest tutaj równa macierzy transponowanej, tę zaś można uzyskać w prosty sposób:

$$\boldsymbol{x} = \boldsymbol{s} \cdot \boldsymbol{W}_c^{-1} = \boldsymbol{s} \cdot \boldsymbol{W}_c^T. \tag{5.66}$$

Podane wyżej zależności można wykazać w sposób analityczny. Elementy dowolnego wiersza macierzy $\boldsymbol{W}_c$ można zapisać następująco:

$$[w_{k,}] = c(k) \left[\cos\left(\frac{k\pi}{2N}\right), \cos\left(\frac{3k\pi}{2N}\right), ..., \cos\left(\frac{(2N-1)k\pi}{2N}\right) \right], \tag{5.67}$$

gdzie:

$$c(k) = \begin{cases} \sqrt{1/N} & \text{dla} \quad k = 0 \\ \sqrt{2/N} & \text{dla} \quad k \neq 0 \end{cases} . \tag{5.68}$$

Funkcje w nawiasie kwadratowym są kosinusoidami o częstotliwościach $k/2N$. Wartości tych funkcji tworzą pewien wektor v_k zawierający N współrzędnych. Pomijając współczynnik $c(k)$, można wykazać, że pomiędzy dwoma dowolnymi wektorami v_l, v_k, $l, k = 0, 1, ..., N-1$ zachodzi związek:

$$\langle v_l, v_l \rangle = \begin{cases} 0 & \text{dla} \quad k \neq l \\ A & \text{dla} \quad k = l \end{cases} , \tag{5.69}$$

gdzie A jest pewną ustaloną wartością.

Formuła (5.69) jest warunkiem ortogonalności wektorów. Rozwijając iloczyn skalarny (5.69) wektorów v_l oraz v_k do pełnej formy, otrzymujemy:

$$\langle v_l, v_k \rangle = \sum_{n=0}^{N-1} \cos\left(\frac{(2n+1)k\pi}{2N}\right) \cdot \cos\left(\frac{(2n+1)l\pi}{2N}\right). \tag{5.70}$$

Odpowiednie związki łatwiej zauważyć, jeśli wzór (5.70) zapisać inaczej, korzystając z tożsamości trygonometrycznej rozstrzygającej, że $2\cos\alpha\cos\beta = \cos(\alpha+\beta)+\cos(\alpha-\beta)$. Wtedy:

$$\langle v_l, v_k\rangle = \frac{1}{2}\sum_{n=0}^{N-1}\cos\left(\frac{(2n+1)(k+l)\pi}{2N}\right) + \frac{1}{2}\sum_{n=0}^{N-1}\cos\left(\frac{(2n+1)(k-l)\pi}{2N}\right). \tag{5.71}$$

Można również zauważyć, że [37]:

$$\sum_{n=0}^{N-1}\cos\left(\frac{(2n+1)a\pi}{2N}\right) = \begin{cases} N & \text{dla} \quad a = 0 \\ -N & \text{dla} \quad a = 2N \\ 0 & \text{dla pozostałych przypadków} \end{cases}. \tag{5.72}$$

Analizując łącznie formuły (5.71) i (5.72), można zaobserwować, że pomiędzy wektorami v_l, v_k zachodzą następujące zależności:

1. jeśli $k \neq l$, to $\langle v_l, v_k\rangle = 0$,
2. jeśli $k = l$ i $k = 0$, to $\langle v_l, v_k\rangle = N$,
3. jeśli $k = l$ i $k \neq 0$, to $\langle v_l, v_k\rangle = N/2$,

co potwierdza, że wektory v_l oraz v_k są ortogonalne dla $l, k = 0, 1, ..., N-1$. Wykazano zatem, że macierz $\boldsymbol{W}_c$ jest ortogonalna.

Rysunek 4 przedstawia pierwszych osiem funkcji bazowych stosowanych w transformacji kosinusowej z zaznaczonymi punktami wartości dyskretnych. Wartości elementów macierzy $\boldsymbol{W}_c$ można odnaleźć na wykresach funkcji bazowych. Można także zauważyć, że wraz ze wzrostem n wzrasta częstotliwość przebiegu kolejnych funkcji.

Dla $N = 8$ macierz jądrowa transformacji kosinusowej $\boldsymbol{W}_c$ ma postać:

$$\boldsymbol{W}_c = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.35356 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}. \tag{5.73}$$

Wiersze macierzy $\boldsymbol{W}_c$ tworzą ortonormalną bazę przestrzeni $\boldsymbol{R}^8$.
Elementy każdego wiersza k macierzy $\boldsymbol{W}_c$ mogą być utożsamiane z próbkami pobranymi w odstępach czasu $t = 0, 1, ..., N-1$ z ciągłego sygnału kosinusoidalnie zmiennego o przebiegu $c(k) \cdot \cos\left(\frac{\pi k(2t+1)}{2N}\right)$.

Przykład 5.6. *Wyznaczyć współczynniki transformaty kosinusowej wektora danych wejściowych* $\boldsymbol{x} = [1, 2, 3, 4]$. *Na podstawie widma odtworzyć dane pierwotne. Otrzymujemy*:

$$\boldsymbol{s} = \boldsymbol{x} \cdot \boldsymbol{W}_c = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0.5000 & 0.5000 & 0.5000 & 0.5000 \\ 0.6533 & 0.2706 & -0.2706 & -0.6533 \\ 0.5000 & -0.5000 & -0.5000 & 0.5000 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix} =$$

$$= \begin{bmatrix} 4.3890 & -3.0719 & 1.0719 & -0.3890 \end{bmatrix}.$$

Na podstawie widma można odtworzyć sygnał pierwotny:

$$\boldsymbol{x} = \boldsymbol{s} \cdot \boldsymbol{W}_c^T = \begin{bmatrix} 4.3890 & -3.0719 & 1.0719 & -0.3890 \end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix} 0.5000 & 0.6533 & 0.5000 & 0.2706 \\ 0.5000 & 0.2706 & -0.5000 & -0.6533 \\ 0.5000 & -0.2706 & -0.5000 & 0.6533 \\ 0.5000 & -0.6533 & 0.5000 & -0.2706 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}.$$

Prostą i odwrotną jednowymiarową transformację kosinusową można wykonać,

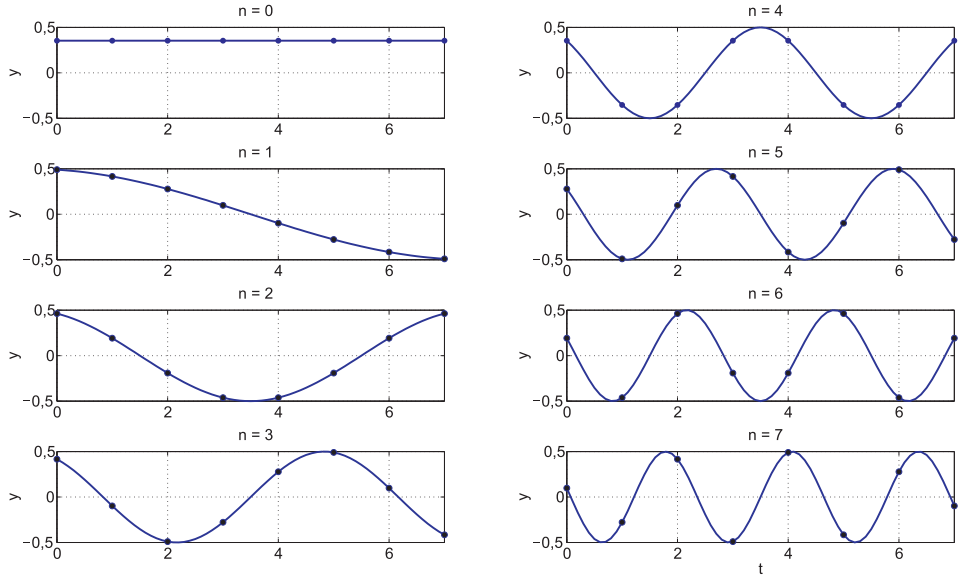
korzystając w tym celu z wymienionej już funkcji Matlaba **dctmtx**(N), którą znajduje się w bibliotece Image Processing Toolobox. Funkcja **dctmtx**(N) generuje opisywaną wcześniej macierz jądrową $\boldsymbol{W}_c$. Zamiast obliczeń, jak w powyższym przykładzie, piszemy prosty program, uzyskując te same wartości współczynników widmowych. Identyczne rezultaty obliczeń otrzymamy, stosując funkcję **dct**($\boldsymbol{x}$), gdzie $\boldsymbol{x}$ jest wektorem danych pierwotnych.

Program 5.7. *Prosta i odwrotna transformacja kosinusowa w Matlabie. Ponieważ mnożenie macierzy jest nieprzemienne, widmo można wyznaczyć na wiele sposobów.*

```
X=[1 2 3 4];          % Wektor danych o dowolnej dlugosci
N=length(X);          % Wyznaczenie dlugosci wektora danych
W=dctmtx(N);          % Macierz jadrowa NxN
S0=X*W                % Prosta DCT (widmo)
X0=S0*W'              % Odwrotna DCT (odtworzenie probek sygnalu)
% Inne wersje obliczen
S1=W*X'
X1=W'*S1
S2=dct(X)             % DCT z funkcji Matlaba dct()
X2=idct(S2)           % Odwrotna DCT z funkcji Matlaba idct()
```

W praktyce odtworzone wartości mogą nieznacznie różnić się od oryginalnych wartości próbek, ze względu na skończoną prezycję obliczeń komputera oraz oprogramowania, za pomocą którego przeprowadza się rachunki.

Wykonywanie obliczeń w przedstawiony powyżej sposób nie jest zbyt często praktykowane ze względu na niekorzystną złożoność obliczeniową, która podobnie jak dla klasycznej transformacji FFT wynosi $O(N^2)$. Ze względu na podobieństwo transformacji DCT do transformacji FFT proces obliczeń można zorganizować tak, aby uzyskać szybką transformację kosinusową o mniejszej złożoności obliczeniowej.



Rys. 4. Wykres pierwszych $N = 8$ funkcji bazowych transformacji kosinusowej $y = c(n)\cos(\frac{\pi n(2t+1)}{2N})$ wraz z punktami reprezentacji dyskretnej

5.3.2. Jednowymiarowa szybka transformacja kosinusowa

Algorytmy szybkich transformacji kosinusowych są stosowane między innymi w kompresji obrazów ruchomych (MPEG), obrazów nieruchomych (JPEG) i w kompresji dźwięku. Algorytmy te mogą mieć postać zarówno rozwiązań sprzętowych, jak i programowych. Standardową prostą i odwrotną transformację kosinusową zdefiniowano za pomocą wzorów (5.60) oraz (5.61). Macierzową wersję wyznaczania współczynników widmowych, którą opisuje wzór (5.64), cechuje złożoność $O(N^2)$. Złożoność tę można jednak zredukować. Zadanie to może być bowiem zrealizowane w czasie $O(N \log_2 N)$, co w efekcie prowadzi do wersji szybkiej algorytmu DCT. Do obliczeń DCT można zastosować opisywany już algorytm FFT. Jeśli dysponujemy programową wersją algorytmu FFT, to jej modyfikacja na potrzeby DCT może być bardziej opłacalna niż projektowanie niezależnych algorytmów o teoretycznie mniejszej złożoności [11,34].

Równanie prostej transformacji kosinusowej (5.60) można zapisać inaczej, grupując oddzielnie parzyste i nieparzyste próbki sygnału wejściowego $x(n)$ [28,40]. Warunkiem nowego zapisu jest równoczesna zmiana sposobu numeracji próbek sygnału wejściowego:

$$s(k) = c(k) \cdot \cdot \left[\sum_{n=0}^{\frac{N}{2}-1} x(2n) \cos\left(\frac{\pi k(4n+1)}{2N} \right) + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) \cos\left(\frac{\pi k(4n+3)}{2N} \right) \right]. \tag{5.74}$$

Oryginalne próbki $x(n)$ mają kolejne numery $n = 0, 1, ..., N/2-1$, a ich nowa numeracja odbywa się według zasady $y(n) = x(2n)$ i $y(N-n-1) = x(2n+1)$, wzór (5.74), po połączeniu obu sum, przyjmuje postać:

$$s(k) = c(k) \sum_{n=0}^{N-1} y(n) \cos\left(\frac{\pi k(4n+1)}{2N} \right). \tag{5.75}$$

Wiadomo, że dyskretną, prostą transformację Fouriera opisuje zależność:

$$s(k) = \sum_{n=0}^{N-1} y(n) \cdot e^{-j2\pi \frac{n}{N} k} = \sum_{n=0}^{N-1} y(n) \left[\cos\left(\frac{2\pi nk}{N} \right) - j \sin\left(\frac{2\pi nk}{N} \right) \right]. \tag{5.76}$$

Jeśli obie strony równania (5.76) pomnożyć przez:

$$e^{-j\pi k/2N} = \cos\left(\frac{\pi k}{2N} \right) - j \sin\left(\frac{\pi k}{2N} \right), \tag{5.77}$$

co jest operacją dozwoloną, to równanie (5.76) przyjmie postać:

$$s(k) \cdot e^{-j\pi k/2N} = \sum_{n=0}^{N-1} y(n) \cdot e^{-j\pi k/2N} \left[\cos\left(\frac{2\pi nk}{N} \right) - j \sin\left(\frac{2\pi nk}{N} \right) \right]. \tag{5.78}$$

Pozostawiając tylko rzeczywistą część równania (5.78), otrzymujemy:

$$\begin{aligned} &Re\left[s(k)\cdot e^{-j\pi k/2N}\right] = \\ &= \sum_{n=0}^{N-1} y(n)\cdot\left[\cos\left(\frac{2\pi nk}{N}\right)\cdot\cos\left(\frac{\pi k}{2N}\right) - \sin\left(\frac{2\pi nk}{N}\right)\cdot\sin\left(\frac{\pi k}{2N}\right)\right]. \end{aligned} \tag{5.79}$$

Po uwzględnieniu zależności trygonometrycznej rozstrzygającej o sumie kątów kosinusa $\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$, ostateczna forma równania (5.79) ma postać:

$$Re\left[s(k)\cdot e^{-j\pi k/2N}\right] = \sum_{n=0}^{N-1} y(n)\cos\left(\frac{\pi k(4n+1)}{2N}\right). \tag{5.80}$$

Porównując DCT w postaci równania (5.75) z równaniem (5.80), będącym rzeczywistą częścią FFT, możemy zapisać, że:

$$s(k) = Re\left[\sum_{n=0}^{N-1} c(k)\cdot y(n)\cdot e^{-j\pi k/2N}\right]. \tag{5.81}$$

Wzór (5.81) jest więc N punktową, dyskretną transformacją Fouriera przeprowadzoną na ciągu danych $y(n)$, utworzonym z przenumerowanych odpowiednio oryginalnych próbek $x(n)$. Dyskretną transformację Fouriera można zaprogramować w wersji FFT, o czym była już mowa.

Oczywistą techniką znalezienia formuły szybkiej, odwrotnej transformacji kosinusowej IDCT jest wykonanie operacji odwrotnych do podanych powyżej. Otrzymujemy wtedy:

$$x^*(n) = Re\left[\sum_{n=0}^{N-1}\left[c(k)\cdot y(n)\cdot e^{-j\pi k/2N}\right]\cdot e^{j2\pi nk/N}\right]. \tag{5.82}$$

Aby uzyskać oryginalny ciąg danych $x(n)$ w należytym porządku, elementy wynikowego ciągu $x^*(n)$ należy przestawić według reguły: $x(2n) = x^*(n)$, $x(2n+1) = x^*(N-n-1)$, dla $n = 0, ..., N/2-1$, uzyskując w ten sposób pierwotny porządek próbek $x(n)$.

W środowisku Matlab szybką, prostą i odwrotną jednowymiarową transformację kosinusową można więc zaprogramować na podstawie wzorów wyprowadzonych w tym podrozdziale.

Program 5.8. *Szybka, prosta i odwrotna jednowymiarowa transformacja kosinusowa z wykorzystaniem FFT.*

```
% Prosta, szybka transformacja kosinusowa
X=[1 2 3 4 5 6 7 8];       % Wektor danych wejsciowych X
N=length(X);               % Liczba elementow wektora danych
Y=zeros(1,N);
for n=0:(N/2)-1
    Y(n+1)=X(2*n+1);        % Dane wejsciowe w nowym porzadku,
    Y(N-n)=X(2*n+2);        % dane te formuja nowy wektor Y
end
F=fft(Y);                  % FFT wektora Y
S=zeros(1,N);
for n=0:N-1
    if (n==0)
        wsp=sqrt(1/N);   else   wsp=sqrt(2/N);    end
    q=-j*n*pi/(2*N);
% Widmo DCT
    S(n+1)=real(wsp*exp(q)*F(n+1));
end
```

5.3.3. Dwuwymiarowa transformacja kosinusowa

Przekształcenie wielowymiarowe DCT jest separowalne, co oznacza, że odpowiednie transformaty można wyznaczać przez sekwencyjne wykonywanie przekształceń jednowymiarowych w każdym wymiarze. Dwuwymiarową transformację kosinusową można przeprowadzić podobnie jak w przypadku dwuwymiarowej transformacji Fouriera – jako złożenie dwóch kosinusowych transformacji jednowymiarowych.

Definicja 5.9. *Dwuwymiarową dyskretną transformacją kosinusową (ang. 2D Discrete Cosine Transform – 2D-DCT) ciągu* $x(m,n)$, $m = 0,1,...,M-1$, $n = 0,1,...,N-1$ *nazywamy ciąg współczynników rozwinięcia*:

$$s(l,k) = c(l) \cdot c(k) \cdot \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) \cdot \cos\left(\frac{\pi l(2m+1)}{2M}\right) \cdot \cos\left(\frac{\pi k(2n+1)}{2N}\right), \tag{5.83}$$

$l = 0,1,...,M-1$, $k = 0,1,...,N-1$.

Definicja 5.10. *Dwuwymiarową odwrotną transformacją kosinusową (ang. 2D Inverse Discrete Cosine Transform – 2D-IDCT) jest odwzorowanie, które na podstawie widma sygnału* $s(l,k)$, $l = 0,...,M-1$, $k = 0,1,...,N-1$ *odtwarza sygnał pierwotny* $x(m,n)$:

$$x(m,n) = \sum_{l=0}^{M-1} \sum_{k=0}^{N-1} c(l) \cdot c(k) \cdot s(l,k) \cdot \cos\left(\frac{\pi l(2m+1)}{2M}\right) \cdot \cos\left(\frac{\pi k(2n+1)}{2N}\right), \tag{5.84}$$

$m = 0,1,...,M-1, \quad n = 0,1,...,N-1$.

W obydwu powyższych definicjach zachodzą związki:

$$\begin{aligned} c(l) &= \begin{cases} 1/\sqrt{M}, & l = 0 \\ \sqrt{2/M}, & l \neq 0 \end{cases} \\ c(k) &= \begin{cases} 1/\sqrt{N}, & k = 0 \\ \sqrt{2/N}, & k \neq 0 \end{cases} . \end{aligned} \tag{5.85}$$

Z biblioteki Image Processing Matlaba można wybrać funkcję **dct2**($\boldsymbol{Q}$), gdzie $\boldsymbol{Q}$ jest macierzą danych o wymiarach $M \times N$. Funkcja zwraca macierz współczynników $\boldsymbol{S} = [s_{lk}]_{M \times N}$.

Transformacja odwrotna może być wykonana z wykorzystaniem wbudowanej funkcji **idct2**($\boldsymbol{S}$). Dla przypadku dwuwymiarowego dyskretną prostą i odwrotną transformację kosinusową można opisać macierzowo następująco:

$$\boldsymbol{S} = \boldsymbol{W}_a \cdot \boldsymbol{X} \cdot \boldsymbol{W}_b^T, \tag{5.86}$$

$$\boldsymbol{X} = \boldsymbol{W}_a^T \cdot \boldsymbol{S} \cdot \boldsymbol{W}_b, \tag{5.87}$$

gdzie: $\boldsymbol{S}, \boldsymbol{X} \in \boldsymbol{R}^{M \times N}$, $\boldsymbol{W}_a = [w_{k,n}]_{M \times M} = c(k) \cos\left(\frac{\pi k(2n+1)}{2M}\right)$, $c(0) = \sqrt{1/M}$, $c(k) = \sqrt{2/M}$ dla $k > 0$ oraz $\boldsymbol{W}_b = [w_{k,n}]_{N \times N} = c(k) \cos\left(\frac{\pi k(2n+1)}{2N}\right)$ oraz $c(0) = \sqrt{1/N}$, $c(k) = \sqrt{2/N}$ dla $k > 0$.

Jeśli $M = N$, wtedy obliczenia upraszczają się, gdyż $\boldsymbol{W}_a = \boldsymbol{W}_b$.

Program 5.9. *Prosta i odwrotna dwuwymiarowa transformacja kosinusowa. Rozwiązanie programowe dla danych dwuwymiarowych $M \times N$.*

```
A=round(2*rand(5,4));          % A jest macierza o wymiarach M X N
[M,N]=size(A);                 % Wymiary M oraz N macierzy A
widmo_1=dct2(A)                % Widmo 2D z funkcji Matlaba dct2()
oryginal_1=idct2(widmo_1)      % Odtworzenie sygnalu pierwotnego 2D
widmo_2=dctmtx(M)*A*(dctmtx(N))'  % Widmo 2D z funkcji dctmtx()
oryginal_2=dctmtx(M)'*widmo_2*dctmtx(N) % Odtworzenie sygnalu 2D
```

5.4. Dyskretna transformacja sinusowa

Dla określenia tej transformacji stosowany jest akronim DST (ang. *Discrete Sine Transform*). DST jest liniową, odwracalną funkcją $f : \boldsymbol{R}^N \to \boldsymbol{R}^N$, która przekształca ciąg dowolnych liczb $x(0), x(1), ..., x(N-1)$ w inny ciąg liczb $s(0), s(1), ..., s(N-1)$.

5.4.1. Jednowymiarowa transformacja sinusowa

Definicja 5.11. *Jednowymiarową dyskretną transformacją sinusową (ang. Discrete Sine Transform – DST) rzeczywistych wartości $x(n)$, $n = 0, 1, ..., N-1$, nazywamy ciąg rzeczywistych współczynników rozwinięcia sygnału $x(n)$:*

$$s(k) = c(k) \sum_{n=0}^{N-1} x(n) \sin\left(\frac{\pi(2n+1)(k+1)}{2N}\right), \qquad k = 0, 1, ..., N-1, \tag{5.88}$$

gdzie:

$$c(k) = \begin{cases} \sqrt{1/N} & dla \quad k = N-1 \\ \sqrt{2/N} & dla \quad k \neq N-1 \end{cases} . \tag{5.89}$$

Definicja 5.12. *Jednowymiarową dyskretną odwrotną transformacją sinusową (ang. Inverse Discrete Sine Transform – IDST) jest odwzorowanie, które na podstawie widma sygnału $s(k)$, $k = 0, 1, ..., N-1$, odtwarza wartości próbek $x(n)$, $n = 0, 1, ..., N-1$:*

$$x(n) = c(k) \sum_{k=0}^{N-1} s(k) \sin\left(\frac{\pi(2n+1)(k+1)}{2N}\right), \qquad n = 0, 1, ..., N-1, \tag{5.90}$$

gdzie:

$$c(k) = \begin{cases} \sqrt{1/N} & dla \quad k = N-1 \\ \sqrt{2/N} & dla \quad k \neq N-1 \end{cases} . \tag{5.91}$$

Z definicji transformacji sinusowych wynika, że jądro transformacji jest takie samo, jak jądro transformacji odwrotnej, co jest udogodnieniem ułatwiającym organizację obliczeń. Macierz jądrowa transformacji (5.88) ma postać:

$$\boldsymbol{W}_s = [w_{k,n}]_{N \times N} = c(k) \sin\left(\frac{\pi(2n+1)(k+1)}{2N}\right), \tag{5.92}$$

gdzie współczynnik $c(k)$ ma znaczenie jak poprzednio.

Ponieważ transformacja jest liniowa, można ją zapisać macierzowo:

$$\boldsymbol{s} = \boldsymbol{x} \cdot \boldsymbol{W}_s. \tag{5.93}$$

Z ortogonalności macierzy $\boldsymbol{W}_s$ wynika, że:

$$\boldsymbol{W}_s^{-1} = \boldsymbol{W}_s^T, \tag{5.94}$$

co ułatwia przeprowadzenie przekształcenia odwrotnego, gdyż macierz odwrotna jest tutaj równa macierzy transponowanej:

$$\boldsymbol{x} = \boldsymbol{s} \cdot \boldsymbol{W}_s^{-1} = \boldsymbol{s} \cdot \boldsymbol{W}_s^T. \tag{5.95}$$

Podane powyżej zależności można wykazać w sposób analityczny, podobnie jak zrobiono to w przypadku transformacji kosinusowej.

Elementy dowolnego wiersza macierzy $\boldsymbol{W}_s$ można zapisać następująco [37]:

$$[w_{k,}] = c(k)\left[\sin\left(\frac{(k+1)\pi}{2N}\right), \sin\left(\frac{3(k+1)\pi}{2N}\right), ..., \sin\left(\frac{(2N-1)(k+1)\pi}{2N}\right)\right].$$

Poszczególne funkcje w nawiasach kwadratowych są sinusoidami o częstotliwościach $(k+1)/(2N)$. Dla danego k wartości tych funkcji tworzą pewien wektor v_k, zawierający N współrzędnych.

Wykażemy, że dla dowolnych dwóch wektorów v_l,v_k, $l, k = 0, 1, ..., N-1$ zachodzi:

$$\langle v_l, v_k \rangle = \begin{cases} 0 & \text{dla} \quad k \neq l \\ A & \text{dla} \quad k = l \end{cases}, \tag{5.96}$$

gdzie A jest pewną ustaloną wartością. Formuła (5.96) jest warunkiem ortogonalności wektorów. Rozwijając iloczyn skalarny (5.96) wektorów v_l oraz v_k do pełnej formy, otrzymujemy:

$$\langle v_l, v_k \rangle = \sum_{n=0}^{N-1} \sin\left(\frac{(2n+1)(l+1)\pi}{2N}\right) \cdot \sin\left(\frac{(2n+1)(k+1)\pi}{2N}\right). \tag{5.97}$$

Odpowiednie związki łatwiej będzie zauważyć, jeśli wzór (5.97) zostanie zapisany inaczej, z wykorzystaniem tożsamości trygonometrycznej rozstrzygającej, że $2\sin\alpha\sin\beta = \cos(\alpha-\beta) - \cos(\alpha+\beta)$. Wtedy [37]:

$$\langle v_l, v_k \rangle = \frac{1}{2}\sum_{n=0}^{N-1} \cos\left(\frac{(2n+1)(l-k)\pi}{2N}\right) - \frac{1}{2}\sum_{n=0}^{N-1} \cos\left(\frac{(2n+1)(l+k+2)\pi}{2N}\right). \tag{5.98}$$

Przywołując ponownie formułę (5.72) oraz analizując formułę (5.98), wnioskujemy, że:

1. jeśli $k \neq l$, to $\langle v_l, v_k \rangle = 0$,
2. jeśli $k = l$ i $k = N-1$, to $\langle v_l, v_k \rangle = N$,
3. jeśli $k = l$ i $k \neq N-1$, to $\langle v_l, v_k \rangle = N/2$,

co potwierdza, że wektory v_l oraz v_k są ortogonalne dla $l, k = 0, 1, ..., N-1$. Wykazano więc, że macierz $\boldsymbol{W}_s$ jest ortogonalna.

Rysunek 5 przedstawia pierwszych osiem funkcji bazowych transformacji sinusowej, wraz z zaznaczonymi punktami wartości dyskretnych. Dyskretne wartości każdego wiersza macierzy $\boldsymbol{W}_s$ są reprezentowane wskaźnikiem $n = 0, ..., 7$. Wartości elementów macierzy $\boldsymbol{W}_s$ można skorelować z danymi na wykresach funkcji bazowych. Czytelnik może łatwo odnaleźć odpowiednie elementy macierzy i skonfrontować je z danym miejscem na wykresie.

Można również zauważyć, że wraz ze wzrostem n wzrasta częstotliwość przebiegu kolejnych funkcji.

Dla $N = 8$ macierz jądrowa transformacji sinusiowej $\boldsymbol{W}_s$ ma postać:

$$\boldsymbol{W}_s = \begin{bmatrix} 0.0975 & 0.2778 & 0.4157 & 0.4904 & 0.4904 & 0.4157 & 0.2778 & 0.0975 \\ 0.1913 & 0.4619 & 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 \\ 0.2778 & 0.4904 & 0.0975 & -0.4157 & -0.4157 & 0.0975 & 0.4904 & 0.2778 \\ 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 \\ 0.4157 & 0.0975 & -0.4904 & 0.2778 & 0.2778 & -0.4904 & 0.0975 & 0.4157 \\ 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 & 0.1913 & -0.4619 \\ 0.4904 & -0.4157 & 0.2778 & -0.0975 & -0.0975 & 0.2778 & -0.4157 & 0.4904 \\ 0.3536 & -0.3536 & 0.3536 & -0.3536 & 0.3536 & -0.3536 & 0.3536 & -0.3536 \end{bmatrix} \tag{5.99}$$

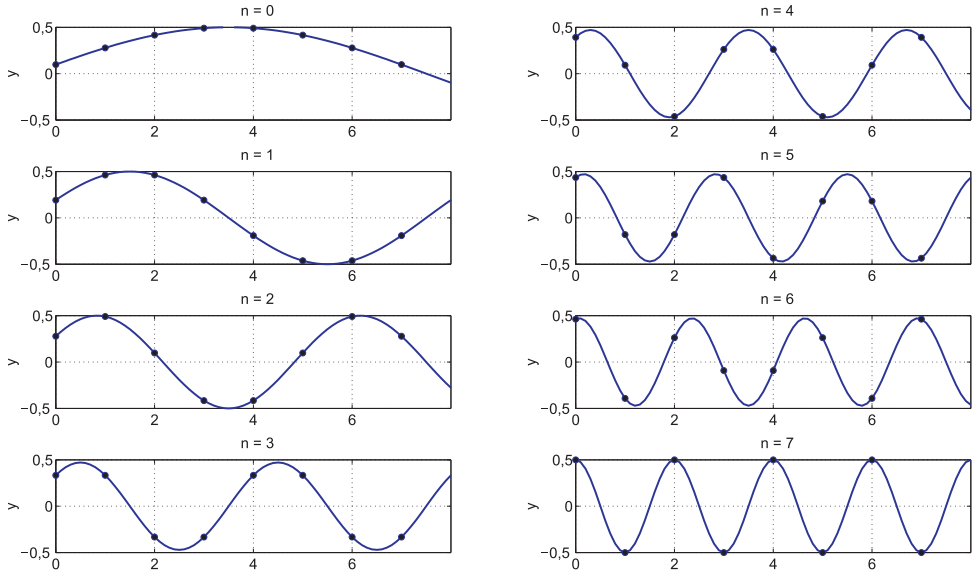
Przykład 5.7. *Wyznaczyć współczynniki widmowe transformaty sinusowej wektora* $\boldsymbol{x} = [1, 2, 3, 4]$. *Potem, na podstawie widma, odtworzyć dane pierwotne.*

$$\boldsymbol{s} = \boldsymbol{x} \cdot \boldsymbol{W_s} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0.2706 & 0.6533 & 0.6533 & 0.2706 \\ 0.5000 & 0.5000 & -0.5000 & -0.5000 \\ 0.6533 & -0.2706 & -0.2706 & 0.6533 \\ 0.5000 & -0.5000 & 0.5000 & -0.5000 \end{bmatrix} =$$

$$= \begin{bmatrix} 4.6194 & -2.0000 & 1.9134 & -1.0000 \end{bmatrix}.$$

Na podstawie wartości współczynników widmowych można odtworzyć dyskretne wartości sygnału pierwotnego:

$$\boldsymbol{x} = \boldsymbol{s} \cdot (\boldsymbol{W}_s)^{-1} = \boldsymbol{s} \cdot \boldsymbol{W}_s^T = \begin{bmatrix} 4.6194 & -2.0000 & 1.9134 & -1.0000 \end{bmatrix} \cdot$$
$$\cdot \begin{bmatrix} 0.2706 & 0.5000 & 0.6533 & 0.5000 \\ 0.6533 & 0.5000 & -0.2706 & -0.5000 \\ 0.6533 & -0.5000 & -0.2706 & 0.5000 \\ 0.2706 & -0.5000 & 0.6533 & -0.5000 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}.$$



Rys. 5. Wykres pierwszych $N = 8$ funkcji bazowych transformacji sinusowej $y = c(k)\sin(\frac{\pi(2n+1)(k+1))}{2N})$ wraz z punktami reprezentacji dyskretnej

Należy mieć na uwadze fakt, że obliczenia DST w Matlabie prowadzone są na podstawie innych definicji. Transformacja sinusowa wykonywana jest przez wywołanie funkcji $\mathbf{dst}(\boldsymbol{x})$, gdzie $\boldsymbol{x}$ jest wektorem danych pierwotnych:

$$s(k) = \sum_{n=1}^{N} x(n) \sin\left(\frac{\pi k n}{N+1}\right), \qquad k = 1, ..., N, \tag{5.100}$$

a sygnał pierwotny odtwarzany jest na podstawie zależności:

$$x(k) = \frac{2}{N+1} \sum_{n=1}^{N} s(n) \sin\left(\frac{\pi k n}{N+1}\right), \qquad k = 1, ...N. \tag{5.101}$$

Zależność (5.101) realizowana jest przez wywołanie funkcji Matlaba **idst**($\boldsymbol{s}$), gdzie $\boldsymbol{s}$ jest wektorem współczynników DST. Ze względu na inny sposób obliczeń wyniki powyższego przykładu będą się różnić od wyników z Matlaba.

Program 5.10. *Prosta i odwrotna transformacja sinusowa w Matlabie.*

```
X=[1 2 3 4];            % Wektor danych o dowolnej dlugosci
N=length(X);            % Wyznaczenie dlugosci wektora danych
MS=zeros(N,N);
for n=0:N-1
    for k=0:N-1
        if (k==N-1);
           wsp=sqrt(1/N);
        else
           wsp=sqrt(2/N);
        end
% Macierz jadrowa N X N
        MS(k+1,n+1)=wsp*sin(pi*(2*n+1)*(k+1)/(2*N));
    end
end
S=X*MS;                 % Widmo transformacji sinusowej
XS=S*MS';               % Odwrotna DST (odtworzenie probek sygnalu)
% Inna wersja obliczen - zgodna z Matlabem
S1=dst(X);              % DST z funkcji Matlaba dst()
X1=idst(S1);            % Odwrotna DST z funkcji Matlaba idst()
```

Podobnie jak w przypadku transformacji kosinusowej, występuje wiele odmian transformacji sinusowej [37]. Inne definicje transformacji sinusowej wprowadzają pewne udogodnienia obliczeniowe, co może przyspieszyć rachunki za-

równo dla prostej, jak i odwrotnej transfomacji. Jeśli prostą transformację sinusową zdefiniować następująco:

$$s(k) = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N-1} x(n) \sin\left(\frac{\pi(n+1)(k+1)}{N+1}\right), \quad k = 0, 1, ..., N-1, \quad (5.102)$$

a transformację odwrotną zapisać w postaci:

$$x(n) = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N-1} s(k) \sin\left(\frac{\pi(n+1)(k+1)}{N+1}\right), \quad n = 0, 1, ..., N-1, \quad (5.103)$$

to jądrowa macierz ma budowę: $\boldsymbol{W}_s = [w_{k,n}]_{N\times N} = \sqrt{\frac{2}{N+1}} \sin\left(\frac{\pi(n+1)(k+1)}{N+1}\right)$, co w świetle prowadzonych już poprzednio wywodów nie jest zaskoczeniem. Nowa definicja transformacji sinusowej zapewnia jednak, że:

$$\boldsymbol{W}_s^{-1} = \boldsymbol{W}_s^T = \boldsymbol{W}_s \ , \quad (5.104)$$

a więc macierzowa wersja odwrotnej transformacji sinusowej nie wymaga operacji transponowania, a tym bardziej odwracania macierzy $\boldsymbol{W}_s$, co sprawia, że można ujednolicić, a tym samym uprościć algorytmy wyznaczania widma oraz odtwarzania dyskretnych punktów sygnału pierwotnego.

5.4.2. Jednowymiarowa szybka transformacja sinusowa

Macierzowa wersja wyznaczania współczynników widmowych opisana wzorem (5.90) ma złożoność obliczeniową $O(N^2)$, co było dyskutowane w rozdziałach opisujących dyskretną transformację kosinusową i transformację Fouriera. Liczbę niezbędnych operacji można jednak zredukować i uzyskać mniejszą złożoność $O(N \log_2 N)$, co prowadzi do wersji szybkiej algorytmu DST.

W poprzednim rozdziale pokazano, jak za pomocą FFT można zrealizować szybką DCT. Wyznaczenie współczynników widmowych sprowadzało się w takim przypadku do odpowiedniego uporządkowania danych wejściowych przed i po operacji FFT. W transformacji kosinusowej bazą przekształcenia są funkcje kosinus, a w transformacji sinusowej odpowiednio dobrane funkcje sinus.

Pomiędzy tymi funkcjami zachodzi oczywisty związek $\cos(\alpha) = \sin(\frac{\pi}{2} \pm \alpha)$, co oznacza, że szybką transformację sinusową można wyprowadzić z szybkiej transformacji kosinusowej, gdzie wykorzystuje się, jak pokazano w rozdziale poprzednim, algorytm FFT. Innymi słowy, można wykazać, że z transformacji (5.60) można wyprowadzić transformację (5.88).

Przypominając znaną już z poprzednich rozważań jednowymiarową transformację kosinusową (5.60):

$$s(k) = c(k) \cdot \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi k(2n+1)}{2N}\right), \qquad k = 0, 1, ..., N-1, \quad (5.105)$$

zmieńmy numerację indeksu k na $N-1-k$. Dane będą więc uporządkowane w odwrotnej kolejności:

$$s(N-1-k) = c(N-1-k) \cdot \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(N-1-k)(2n+1)}{2N}\right). \quad (5.106)$$

Wykonując mnożenie i grupowanie zmiennych, funkcję kosinus we wzorze (5.106) można zapisać inaczej:

$$\cos\left(\frac{\pi(N-1-k)(2n+1)}{2N}\right) =$$

$$= \cos\left(\frac{-\pi(2n+1)(k+1) + \pi N(2n+1)}{2N}\right) = \cos\left(\frac{\pi}{2} + n\pi - \frac{(2n+1)(k+1)\pi}{2N}\right). \quad (5.107)$$

Ponieważ $\cos(\alpha) = \sin(\frac{\pi}{2} - \alpha)$, wzór (5.107) można zapisać w postaci tożsamości:

$$\cos\left(\frac{\pi}{2} + n\pi - \frac{(2n+1)(k+1)\pi}{2N}\right) = \sin\left(\frac{(2n+1)(k+1)\pi}{2N} - n\pi\right). \quad (5.108)$$

Przywołując z kolei wzór trygonometryczny na różnicę kątów funkcji sinus $\sin(\alpha - \beta) = \sin\alpha\cos\beta - \cos\alpha\sin\beta$ oraz wiedząc, że $\cos(n\pi) = (-1)^n$, otrzymujemy nową postać wzoru (5.108):

$$\sin\left(\frac{(2n+1)(k+1)\pi}{2N} - n\pi\right) = (-1)^n \sin\left(\frac{(2n+1)(k+1)\pi}{2N} - n\pi\right). \tag{5.109}$$

Ostatecznie, po zmianie zasad indeksowania wskaźnika k, jednowymiarowa transformacja kosinusowa (5.105) może być zapisana w postaci równoważnej:

$$s(N-1-k) = c(N-1-k) \cdot \sum_{n=0}^{N-1} x(n)(-1)^n \sin\left(\frac{\pi(2n+1)(k+1)}{2N}\right), \tag{5.110}$$

co oznacza, że porządkując dane wejściowe i wykonując na tych danych DCT, w efekcie otrzymujemy DST.

Pozwala to zaprojektować szybki algorytm transformacji sinusowej, którego złożoność wynosi $O(N \log_2 N)$.

Program 5.11. *Szybka, prosta i odwrotna jednowymiarowa transformacja sinusowa, gdzie w rachunkach wykorzystywana jest DCT oraz FFT.*

```
% Prosta, szybka transformacja sinusowa
X=[1 2 3 4 5 6 7 8];          % Wektor danych wejsciowych X
N=length(X);                  % Liczba elementow wektora danych
Y=zeros(1,N);
for n=0:N-1
    Y(n+1)=(-1)^n*X(n+1);     % Nowy porzadek danych
end
X=Y;
for n=0:N/2-1                 % DCT z wykorzystaniem FFT
    Y(n+1)=X(2*n+1);      Y(N-n)=X(2*n+2);
end
A=zeros(1,N);              S=zeros(1,N);
F=fft(Y);
for n=0:N-1
    if (n==0) wsp=sqrt(1/N);  else  wsp=sqrt(2/N);  end
    q=-j*n*pi/(2*N);       A(n+1)=real(wsp*exp(q)*F(n+1));
end
```

```
for n=0:N-1
    S(n+1)=A(N-n);          % Ostateczne uporzadkowanie widma
end
% Odwrotna, szybka transformacja sinusowa
S=[1 2 3 4 5 6 7 8];         % Wektor danych wejsciowych (widmo)
N=length(S);                 % Liczba elementow wektora danych
A1=zeros(1,N); A=zeros(1,N);
for n=0:N-1
    A(n+1)=S(N-n);          % Nowy porzadek danych
end
for n=0:N-1                 % iDCT z wykorzystaniem iFFT
    if (n==0)
       wsp=sqrt(1/N);
    else
       wsp=sqrt(2/N);
    end
    q=j*n*pi/(2*N);
    A1(n+1)=wsp*exp(q)*A(n+1);
end
F=real(ifft(A1));
for n=0:N/2-1
    A1(2*n+1)=F(n+1);
    A1(2*n+2)=F(N-n);
end
% Ostateczne uporzadkowanie danych
for n=0:N-1
    X(n+1)=(-1)^n*A1(n+1);
end
X=N*X;                  % Widmo iDST
```

5.4.3. Dwuwymiarowa transformacja sinusowa

Przekształcenie wielowymiarowe DST jest separowalne, co oznacza, że odpowiednie transformaty można wyznaczać przez sekwencyjne wykonywanie przekształceń jednowymiarowych w każdym wymiarze. Dwuwymiarową transformację sinusową można wykonać w postaci złożenia dwóch sinusowych transformacji jednowymiarowych.

Definicja 5.13. *Dwuwymiarową dyskretną transformacją sinusową (ang. 2D Discrete Sine Transform – 2D-DST) ciągu danych $x(m,n)$, $m = 0,1,...,M-1$, $n = 0,1,...,N-1$ nazywamy ciąg współczynników rozwinięcia*:

$$\begin{aligned} s(l,k) &= c(l)\cdot c(k)\cdot \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x(m,n)\cdot \\ &\cdot \sin\left(\frac{\pi(2m+1)(k+1)}{2M}\right)\cdot \sin\left(\frac{\pi(2n+1)(l+1))}{2N}\right), \\ l &= 0,1,...,M-1, \quad k = 0,1,...,N-1. \end{aligned} \tag{5.111}$$

Definicja 5.14. *Dwuwymiarową odwrotną transformacją sinusową (ang. 2D Inverse Discrete Sine Transform – 2D-IDST) jest odwzorowanie, które na podstawie widma sygnału $s(l,k)$, $l = 0,...,M-1$, $k = 0,1,...,N-1$ odtwarza sygnał pierwotny $x(m,n)$*:

$$\begin{aligned} x(m,n) &= \sum_{l=0}^{M-1}\sum_{k=0}^{N-1} c(l)\cdot c(k)\cdot s(l,k)\cdot \\ &\cdot \sin\left(\frac{\pi(2m+1)(l+1)}{2M}\right)\cdot \sin\left(\frac{\pi(2n+1)(k+1)}{2N}\right), \\ m &= 0,1,...,M-1, \quad n = 0,1,...,N-1. \end{aligned} \tag{5.112}$$

Dla przypadku dwuwymiarowego dyskretną prostą i odwrotną transformację sinusową można przedstawić w zapisie macierzowym:

$$\boldsymbol{S} = \boldsymbol{W_a}\cdot \boldsymbol{X}\cdot \boldsymbol{W}_b^T, \tag{5.113}$$

$$\boldsymbol{X} = \boldsymbol{W}_a^T\cdot \boldsymbol{S}\cdot \boldsymbol{W}_b, \tag{5.114}$$

gdzie: $\boldsymbol{S}, \boldsymbol{X} \in \boldsymbol{R}^{M\times N}$, $\boldsymbol{W_a} = [w_{k,n}]_{M\times M} = c(k)\sin\left(\dfrac{\pi(2n+1)(k+1)}{2M}\right)$,

$\boldsymbol{W_b} = [w_{k,n}]_{N\times N} = c(l)\sin\left(\dfrac{\pi(2n+1)(k+1)}{2N}\right)$, oraz:

$$c(k) = \begin{array}{ll} \sqrt{1/N} & \text{dla} \quad k = N-1 \\ \sqrt{2/N} & \text{dla} \quad k \neq N-1 \end{array}, \quad c(l) = \begin{cases} \sqrt{1/M} & \text{dla} \quad l = M-1 \\ \sqrt{2/M} & \text{dla} \quad l \neq M-1 \end{cases}. \tag{5.115}$$

Jeśli $M = N$, obliczenia się upraszczają, gdyż $\boldsymbol{W}_a = \boldsymbol{W}_b$.

5.5. Funkcje i transformacja Hartleya

Dyskretną transformację Hartleya wprowadził Bracewell w 1983 r. [11]. Dyskretna transformacja Hartleya, podobnie jak transformacja Fouriera, jest, jak się przekonamy, transformacją liniową $\boldsymbol{R}^N \to \boldsymbol{R}^N$, zatem może być również opisywana macierzowo. Transformację Hartleya cechuje wiele podobieństw do transformacji Fouriera, a jej wyróżnikiem są działania na liczbach rzeczywistych. Zaletą tej transformacji jest zdolność przekształcania funkcji o wartościach rzeczywistych w funkcje o wartościach rzeczywistych, a więc inaczej niż w przypadku transformacji Fouriera, gdzie mamy zawsze $\boldsymbol{R}^N \to \boldsymbol{C}^N$. Pomiędzy transformacją Hartleya i transformacją Fouriera zachodzą związki, które zostaną zaprezentowane wraz z odpowiednimi przykładami w Matlabie. Funkcje Hartleya oznaczane jako $Hart(x,t)$ opisane są dwoma wskaźnikami. Wskaźnik x określa numer (rząd) funkcji, a wskaźnik t służy do wyznaczania bieżącej wartości funkcji w przedziale określoności. Uwzględniając zależności trygonometryczne, można zapisać:

$$\begin{aligned} Hart(x,t) &= \cos\left(\frac{2\pi}{T}xt\right) + \sin\left(\frac{2\pi}{T}xt\right) = \\ &= \sqrt{2}\sin\left(\frac{2\pi}{T}xt + \pi/4\right) = \sqrt{2}\cos\left(\frac{2\pi}{T}xt - \pi/4\right), \end{aligned} \tag{5.116}$$

gdzie $x = 0, \pm 1, \pm 2, ...,$.
Funkcje Hartleya są ortogonalne, ale można je normować.

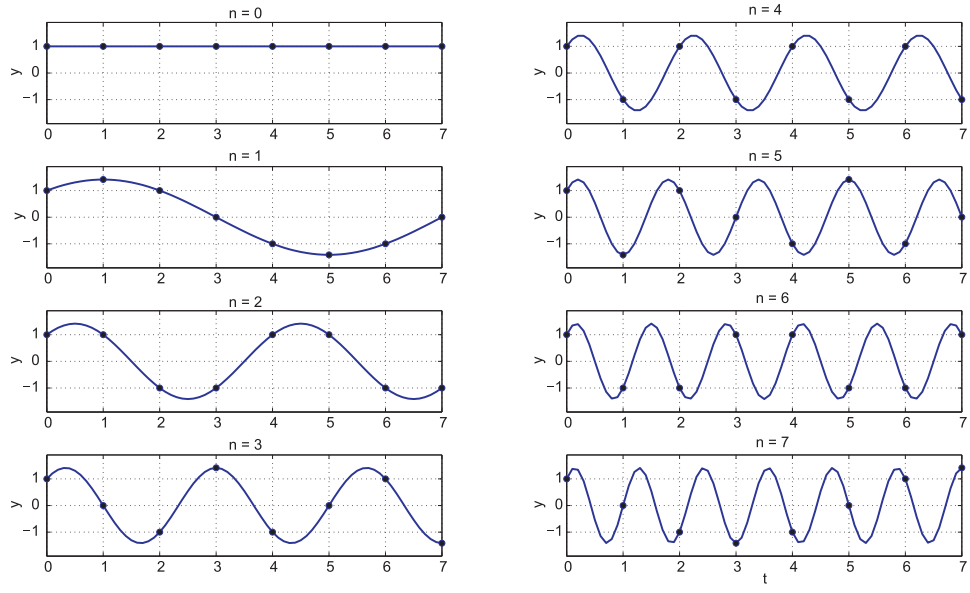
Ortonormalny zbiór $\{h_x(t)\}$ takich funkcji, określony na przedziale $t = [t_0, t_o + T]$, gdzie T jest okresem funkcji, jest zdefiniowany następująco :

$$\{h_x(t)\} = \frac{1}{\sqrt{N}} \left(\cos\left(\frac{2\pi}{T} xt\right) + \sin\left(\frac{2\pi}{T} xt\right) \right), \tag{5.117}$$

wówczas:

$$\langle h_n(t), h_m(t) \rangle = \begin{cases} 1 & n = m \\ 0 & n \neq m \end{cases} . \tag{5.118}$$

Rysunek 6 przedstawia pierwszych osiem ciągłych funkcji Hartleya z zaznaczonymi punktami wartości, których znajomość jest niezbędna w operacjach dyskretnych. Można zauważyć, że wraz ze wzrostem n wzrasta częstotliwość przebiegu kolejnych funkcji Hartleya.



Rys. 6. Wykres pierwszych $N = 8$ funkcji Hartleya $y = \cos(n\frac{2\pi}{N}t) + \sin(n\frac{2\pi}{N}t)$ wraz z punktami reprezentacji dyskretnej

5.5.1. Dyskretna jednowymiarowa transformacja Hartleya

Definicja 5.15. *Jednowymiarową dyskretną transformacją Hartleya (ang. Discrete Hartley Transform – DHT) sygnału $x(n)$, określonego w chwilach $n = 0, 1, ..., N-1$, nazywamy ciąg rzeczywistych współczynników $s(k) \in \boldsymbol{R}^N$ rozwinięcia sygnału $x(n) \in \boldsymbol{R}^N$:*

$$s(k) = \alpha \sum_{n=0}^{N-1} x(n) \left(\cos\left(\frac{2\pi}{N} kn\right) + \sin\left(\frac{2\pi}{N} kn\right) \right), \quad k = 0, 1, ..., N-1. \tag{5.119}$$

Definicja 5.16. *Jednowymiarową odwrotną, dyskretną transformację Hartleya (ang. Inverse Discrete Hartley Transform – IDHT) jest odwzorowanie, które na podstawie próbek $s(k) \in \boldsymbol{R}^N$, $k = 0, ..., N-1$ widma sygnału odtwarza próbki sygnału pierwotnego $x(n) \in \boldsymbol{R}^N$, $n = 0, ..., N-1$:*

$$x(n) = \beta \sum_{k=0}^{N-1} s(k) \left(\cos\left(\frac{2\pi}{N} kn\right) + \sin\left(\frac{2\pi}{N} kn\right) \right), \quad n = 0, 1, ..., N-1. \tag{5.120}$$

Dla zwiększenia przejrzystości formuł suma funkcji trygonometrycznych przedstawionych w powyższych definicjach może być zapisywana w formie skróconej:

$$cas(\frac{2\pi}{N} kn) = \cos\left(\frac{2\pi}{N} kn\right) + \sin\left(\frac{2\pi}{N} kn\right). \tag{5.121}$$

Z definicji transformacji Hartleya wynika, że jądro transformacji jest takie samo, jak jądro transformacji odwrotnej, co jest zaletą ułatwiającą organizację obliczeń. Podobieństwa transformacji Hartleya do transformacji Fouriera można zobaczyć, gdy prostą transfomację Fouriera zapiszemy w postaci:

$$s_f(k) = \alpha \sum_{n=0}^{N-1} x(n) \left(\cos\left(\frac{2\pi}{N} kn\right) - j \sin\left(\frac{2\pi}{N} kn\right) \right), \quad k = 0, 1, ..., N-1. \tag{5.122}$$

Podobieństwa wzorów (5.119) oraz (5.122) są oczywiste, z dokładnością do jednostki urojonej j. W dyskretnej transformacji Fouriera sekwencję liczb rzeczywistych rejestrowanych w dziedzinie czasu przekształca się na sekwencję

liczb zespolonych w dziedzinie częstotliwości. Połowa tak uzyskanych liczb jest jednak nadmiarowa, czyli informacja w widmie się powtarza.

Niech współczynniki widmowe Hartleya mają oznaczenie $s_h(k)$, a współczynniki widmowe Fouriera $s_f(k)$, $k = 0, ..., N-1$. Wtedy prawdziwe są następujące związki:

$$\begin{aligned} Re[s_f(k)] &= Re[s_f(-k)], \\ Im[s_f(k)] &= Im[-s_f(-k)]. \end{aligned} \tag{5.123}$$

Transformacja Hartleya usuwa nadmiarowość transformaty Fouriera, gdyż:

$$s_h(k) = Re[s_f(k)] - Im[s_f(k)]. \tag{5.124}$$

Z przytoczonych zależności wynika, że z transformaty Hartleya można ustalić odpowiednie współczynniki widmowe Fouriera.

Niech N współczynników Hartleya jest uporządkowanych w sposób następujący: $s_h(0), s_h(1), ..., s_h(N-1), s_h(0)$. Jak widać, do ciągu N współczynników na pozycji N została dopisana wartość duplikatu współczynnika $s_h(0)$. Dla takiego porządku danych zachodzą następujące związki pomiędzy współczynnikami widmowymi Hartleya i Fouriera:

$$Re[s_f(k)] = +\frac{1}{2}[s_h(N-k) + s_h(k)], \qquad k = 0, ..., N-1, \tag{5.125}$$

$$Im[s_f(k)] = -\frac{1}{2}[s_h(N-k) - s_h(k)], \qquad k = 0, ..., N-1, \tag{5.126}$$

czyli:

$$Re[s_f(k)] = \sum_{n=0}^{N-1} \cos\left(\frac{2\pi}{N} kn\right), \tag{5.127}$$

$$Im[s_f(k)] = \sum_{n=0}^{N-1} -\sin\left(\frac{2\pi}{N} kn\right). \tag{5.128}$$

co oznacza, że części parzysta i nieparzysta (ze znakiem ujemnym) widma Hartleya są identyczne jak część parzysta i nieparzysta widma Fouriera. W tym sensie, jeśli dane pierwotne przyjmują wartości rzeczywiste, są to transformacje

równoważne, gdyż znając współczynniki widmowe jednej transformaty, można wyznaczyć współczynniki drugiej i *vice versa*. Można zauważyć, że przekształcenie (5.119) można zapisać w równoważnej postaci macierzowej:

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{x} \cdot \boldsymbol{W}_h, \tag{5.129}$$

gdzie: $\boldsymbol{W_h} = [w_{k,n}]_{N\times N} = \left[\cos\left(\frac{2\pi nk}{N}\right) + \sin\left(\frac{2\pi nk}{N}\right)\right]$ jest macierzą jądrową (jądrem) przekształcenia (5.129).
Dla $N = 8$ ortogonalna macierz jądrowa Hartleya $\boldsymbol{W}_h$ ma postać:

$$\boldsymbol{W}_h = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \sqrt{2} & 1 & 0 & -1 & -\sqrt{2} & -1 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & -1 & \sqrt{2} & -1 & 0 & 1 & -\sqrt{2} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\sqrt{2} & 1 & 0 & -1 & \sqrt{2} & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & -1 & -\sqrt{2} & -1 & 0 & 1 & \sqrt{2} \end{bmatrix} \tag{5.130}$$

Można zauważyć, że elementy macierzy (5.130) odpowiadają wybranym wartościom odpowiednich funkcji Hartleya na rys. 6. Na rysunku wartości te oznaczono punktami. Numerując wiersze macierzy (5.130) identycznie jak numery funkcji na rys. 6, otrzymujemy dyskretną reprezentację poszczególnych funkcji Hartleya. W wersji macierzowej przekształcenie (5.120) ma postać:

$$\boldsymbol{x} = \beta \cdot \boldsymbol{s} \cdot \boldsymbol{W}_h, \tag{5.131}$$

a więc jest takie samo, jak (5.129) z dokładnością do czynnika β. Przeważnie przyjmuje się $\alpha = 1$ oraz $\beta = 1/N$.

Przykład 5.8. *Wyznaczyć współczynniki widmowe Hartleya wektora danych* $\boldsymbol{x} = [1, 2, 3, 4]$. *Przyjmujemy* $\alpha = 1$, $\beta = 1/4$. *Dodatkowo sprawdzić związki widma Hartleya z widmem Fouriera.*

Otrzymujemy:

$$\boldsymbol{s}_h = \alpha \cdot \boldsymbol{x} \cdot \boldsymbol{W}_h = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 10 & -4 & -2 & 0 \end{bmatrix}.$$

Widmo Fouriera sygnału wejściowego $\boldsymbol{x}$ *jest następujące:*
$\boldsymbol{s}_f = \begin{bmatrix} 10 & -2+2j & -2 & -2-2j \end{bmatrix}$.
Wykorzystując zależności (5.123) oraz (5.124), można natychmiast sprawdzić wzajemne powiązania widm Hartleya i Fouriera.

Przykład 5.9. *Odtworzyć dyskretny sygnał pierwotny* $\boldsymbol{x}$ *na podstawie wyliczonych w poprzednim przykładzie współczynników widmowych Hartleya* $\boldsymbol{s} = [10, -4, -2, 0]$. *Przyjmując, podobnie jak w poprzednim przykładzie, założenie, że* $\beta = 1/4$, *uzyskujemy*:

$$\boldsymbol{x} = \beta \cdot \boldsymbol{s} \cdot \boldsymbol{W}_h = \frac{1}{4} \cdot \begin{bmatrix} 10 & -4 & -2 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}.$$

Program 5.12. *Program, za pomocą którego na podstawie wzorów definicyjnych można wyznaczyć dyskretne, jednowymiarowe widmo Hartleya wektora* $\boldsymbol{x}$ *oraz z widma* $\boldsymbol{s}$ *odtworzyć wartości danych wejściowych.*

```
% Prosta transformacja Hartleya %
x=[1,2,3,4];              % Dane o wartosciach rzeczywistych
N=length(x);              % Dlugosc wektora danych
sH=zeros(1,N);            % Rezerwacja miejsca na wynik
H=zeros(N,N);             % Przygotowanie macierzy Hartleya
```

```
% Macierz Hartleya H
% Wspolczynniki widmowe Hartleya sH
for k=0:N-1
    for n=0:N-1
        H(n+1,k+1)=cos(2*pi*n*k/N) + sin(2*pi*n*k/N);
        sH(k+1)=sH(k+1)+x(n+1)*H(n+1,k+1);
    end
end
sH                          % Widmo z definicji
x*H                         % To samo widmo w zapisie macierzowym
% Odwrotna transformacja Hartleya %
s=[1,2,3,4];                % Wektor danych wejsciowych (widmo)
N=length(s);                % Dlugosc wektora danych
x=zeros(1,N);               % Rezerwacja miejsca na wynik
for n=0:N-1
    for k=0:N-1
        x(n+1)=x(n+1)+s(k+1)*(cos(2*pi*n*k/N)+sin(2*pi*n*k/N));
     end
end
1/N*x                       % Odtworzenie sygnalu z definicji
1/N*s*H                     % To samo w rachunku macierzowym
```

5.5.2. Dyskretna szybka transformacja Hartleya

Jak już dowiedziono w niniejszym rozdziale, transformacja Hartleya wykazuje podobieństwa do klasycznej transformacji Fouriera. Z tych powodów złożoność czasowa algorytmów realizujących te transformacje jest również podobna. W rozdziale 5.2 poświęconym transformacji Fouriera opisano złożoność tego algorytmu. Podobne rozważania można przeprowadzić dla wyznaczenia złożoności czasowej algorytmu transformacji Hartleya. Identycznie jak w przypadku transformacji Fouriera, metoda bezpośrednia wymaga mnożenia macierzy $\boldsymbol{W}_h$ o rozmiarach $N \times N$ przez N wymiarowy wektor $\boldsymbol{x}$.

Realizacja zadania wymaga więc N^2 zespolonych mnożeń oraz $N \cdot (N-1)$ zespolonych dodawań. Ten sposób realizacji zadania jest więc bardzo kosztowny, szczególnie dla dużych N. Ze względu na opisywane związki pomiędzy obydwiema transformatami do wyznaczenia transformaty Hartleya można bezpośrednio zastosować algorytm szybkiej transformacji Fouriera (FFT), który w postaci funkcji bibliotecznej jest implementowany w wielu pakietach matematycznych, w tym w Matlabie. Algorytm szybkiej transformacji Hartleya z użyciem FFT wykorzystuje zasadę podziału ciągu N danych wejściowych na dwa niezależne podciągi $N/2$ próbek występujących na parzystych i nieparzystych pozycjach ciągu wejściowego. Warunek ten oznacza, że liczba elementów wejściowego ciągu danych musi być potęgą liczby 2. Złożoność obliczeniowa wyniesie wtedy, podobnie jak dla omawianego już algorytmu FFT, $O(N \log_2 N)$.

Program 5.13. *Wyznaczenie widma Hartleya na podstawie znanego wcześniej widma Fouriera i vice versa – na podstawie widma Hartleya można ustalić odpowiadające mu widmo Fouriera i odtworzyć sygnał pierwotny.*

```
% Widmo Harleya z wyznaczonego wczesniej widma Fouriera
x=[1,2,3,4];            % Wejsciowy wektor danych
sF=fft(x);              % Widmo Fouriera
sH=real(sF)-imag(sF);   % Widmo Hartleya
% Widmo Fouriera z wyznaczonego wczesniej widma Hartleya i
N=length(sF);         % Dlugosc sekwencji danych (widmo Hartleya)
sp=zeros(1,N);        % Miejsce na parzysta czesc widma Hartleya
sn=zeros(1,N);        % Miejsce na nieparzysta czesc widma Hartleya
for n=0:N-1
    for k=0:N-1
        sp(n+1)=sp(n+1)+x(k+1)*cos(2*pi*n*k/N);
        sn(n+1)=sn(n+1)+x(k+1)*sin(2*pi*n*k/N);
    end
end
F=sp-j*sn             % Widmo Fouriera jako zlozenie widm Hartleya
x0=ifft(F)            % Odtworzenie sygnalu pierwotnego
```

5.5.3. Dyskretna dwuwymiarowa transformacja Hartleya

Definicja 5.17. *Dwuwymiarową dyskretną transformacją Hartleya (ang. 2D Discrete Hartley Transform - 2D-DHT) danych $x(m,n)$, $m = 0,1,...,M-1$, $n = 0,1,...,N-1$ nazywamy ciąg rzeczywistych współczynników rozwinięcia*:

$$s(l,k) = \alpha \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) \cdot cas\left(\frac{2\pi il}{M} + \frac{2\pi nk}{N}\right), \tag{5.132}$$

gdzie $l = 0,1,...,M-1$, $k = 0,1,...,N-1$.

Definicja 5.18. *Dwuwymiarową odwrotną transformacją Hartleya (ang. 2D Inverse Discrete Hartley Transform - 2D-IDHT) jest odwzorowanie, które na podstawie widma sygnału $s(l,k)$, $l = 0,...,M-1$, $k = 0,1,...,N-1$ odtwarza sygnał pierwotny w punktach $x(m,n)$*:

$$x(m,n) = \beta \sum_{l=0}^{M-1} \sum_{k=0}^{N-1} s(l,k) \cdot cas\left(\frac{2\pi il}{M} + \frac{2\pi nk}{N}\right), \tag{5.133}$$

gdzie $cas(\theta) = \cos(\theta) + \sin(\theta)$, $\quad m = 0,1,...,M-1 \quad i \quad n = 0,1,...,N-1$.

Dla przypadku dwuwymiarowego dyskretną prostą i odwrotną transformację Hartleya można przedstawić w zapisie macierzowym:

$$\boldsymbol{S} = \alpha \cdot \boldsymbol{W}_a \cdot \boldsymbol{X} \cdot \boldsymbol{W}_b, \tag{5.134}$$

$$\boldsymbol{X} = \beta \cdot \boldsymbol{W}_a \cdot \boldsymbol{S} \cdot \boldsymbol{W}_b, \tag{5.135}$$

gdzie: $\boldsymbol{S}, \boldsymbol{X} \in \boldsymbol{R}^{M \times N}$, $\boldsymbol{W}_a = [w_{n,k}]_{N \times N} = \left[\cos\left(\frac{2\pi nk}{N}\right) + \sin\left(\frac{2\pi nk}{N}\right)\right]$,

$\boldsymbol{W}_b = [w_{n,k}]_{M \times M} = \left[\cos\left(\frac{2\pi nk}{M}\right) + \sin\left(\frac{2\pi nk}{M}\right)\right]$.

Wzory (5.132) oraz (5.133) będą wzajemnie odwrotne, jeśli respektowany będzie warunek:

$$\alpha \cdot \beta = \frac{1}{N \cdot M}. \tag{5.136}$$

5.6. Transformacja Gooda–Thomasa

Jeśli liczba N nie jest potęgą liczby 2, to nie można stosować *explicite* opisanych wcześniej metod wyznaczania widma, co wynika z budowy schematu motylkowego. Liczbę N można jednak zapisać inaczej, dzięki czemu, po modyfikacjach, nadal można wykorzystywać opisywane wcześniej transformacje.

Realizację tego zadania umożliwiają dwie grupy algorytmów – algorytm Cooleya–Tukeya, dla $N = N_1 \cdot N_2$, gdzie N_1 oraz N_2 są liczbami naturalnymi, lub algorytm Gooda–Thomasa, w którym liczba N może być przedstawiona jako iloczyn liczb względnie pierwszych – ich jedynym wspólnym dzielnikiem jest liczba 1 [8,40]. Pierwszy z algorytmów opisany został przystępnie między innymi w książce [40]. Drugi algorytm zostanie szczegółowo omówiony poniżej. Idea algorytmu Gooda–Thomasa opiera się na spostrzeżeniu, że liczbę naturalną N można przedstawić jako iloczyn liczb pierwszych – $N = r_1 r_2 ... r_p$, co oznacza, że zamiast pojedynczej klasycznej DFT, której złożoność nie jest korzystna i wynosi $O(N^2)$, można zastosować p mniejszych transformat DFT. Nakład obliczeń można wtedy zmniejszyć do poziomu $O(N(r_1 + ... + r_p))$.

Wymogiem algorytmu Gooda–Thomasa jest znalezienie właściwego uporządkowania danych wejściowych i wyjściowych oraz zbudowanie odpowiedniego diagramu przepływu, na podstawie którego można wyznaczyć transformatę.

Niech próbki analizowanego sygnału wejściowego są oznaczone następująco: $x(n)$, $n = 0, ..., N-1$. Próbki umieszczane są w tablicy $A(n_1, n_2)$, gdzie indeksy n_1 oraz n_2 przyjmują wartości w zakresie $n_1 = 0, ..., N_1 - 1$ oraz $n_2 = 0, ..., N_2 - 1$, a relacje między indeksami tablicy A oraz bieżącym numerem n próbki $x(n)$ są ustalane jako rozwiązanie kongruencji:

$$\begin{array}{ll} n_1 \equiv n \pmod{N_1} & n_1 = 0, 1, ..., N_1 - 1 \\ n_2 \equiv n \pmod{N_2} & n_2 = 0, 1, ..., N_2 - 1 \end{array} . \tag{5.137}$$

Wiersze i kolumny tablicy A będą potem wykorzystywane w diagramie przepływu dla ustalonego N.

Dla takich założeń liczbę n wylicza się z równań:

$$\begin{aligned} n &= n_2 \cdot t \cdot N_2 + n_1 \cdot s \cdot N_1 \qquad (\text{mod } N) \\ N_1 \cdot s + N_2 \cdot t &= 1 \end{aligned} . \tag{5.138}$$

Liczby s oraz t wyznacza się z opisywanego już wcześniej rozszerzonego algorytmu Euklidesa (Program 3.2).

W praktyce, zamiast rozwiązywania układu kongruencji (5.137), wykorzystuje się prostszą, ekwiwalentną formułę przeliczeniową, co wynika z rozważań przeprowadzonych w rozdziale 3:

$$n = n_1 N_2 + n_2 N_1 \qquad (\text{mod } N).$$

Tablica A wypełniana jest elementami ciągu x następująco: $A(n_1, n_2) = x(n)$. Widmo tworzy ciąg wartości $s(k)$, $k = 0, ..., N-1$. Porządek współczynników widmowych wyznaczany jest z formuły:

$$k = k_1 \cdot [N_2^{-1}]_{N_1} \cdot N_1 + k_2 \cdot [N_1^{-1}]_{N_2} \cdot N_2 \qquad (\text{mod } N), \tag{5.139}$$

gdzie zapis $[N_2^{-1}]_{N_1}$ oznacza liczbę odwrotną do N_2 modulo N_1. Podobne znaczenie ma zapis $[N_1^{-1}]_{N_2}$. Liczby odwrotne modulo n wyznaczyć można za pomocą programu (Program 3.1).

Dla przyjętych założeń tablica B wypełniana jest elementami ciągu s następująco: $B(k_1, k_2) = s(k_1 \cdot [N_2^{-1}]_{N_1} + k_2 \cdot [N_1^{-1}]_{N_2}) \pmod N$. Po ustaleniu porządku danych wejściowych i wyjściowych transformacja Gooda–Thomasa opisana jest zależnością:

$$B(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} A(n_1, n_2) W_{N_2}^{n_2 k_2} \right] W_{N_1}^{n_1 k_1}, \tag{5.140}$$

gdzie $W_{N_i}^{n_i k_i} = e^{-j2\pi \frac{n_i}{N_i} k_i}$, $i = 1, 2$.
Transformacja odwrotna przeprowadzana jest według formuły:

$$A(n_1, n_2) = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} B(k_1, k_2) W_{N_2}^{-n_2 k_2} \right] W_{N_1}^{-n_1 k_1}, \tag{5.141}$$

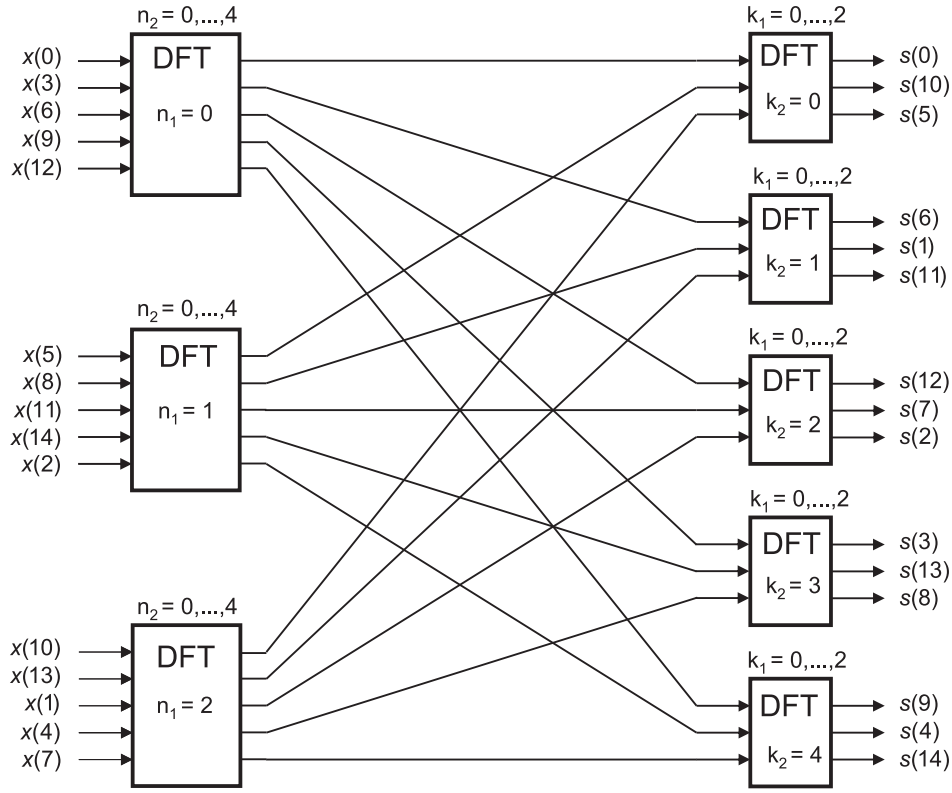
gdzie porządek elementów ciągu wejściowego (teraz widma) jest ustalany identycznie jak w transformacji prostej.

Sposób transformowania danych w transformacji Gooda–Thomasa przedstawia diagram przepływowy zaprezentowany na rys. 7 dla przypadku, gdy liczba próbek sygnału wejściowego wynosi $N = 15$ oraz $N_1 = 3$, $N_2 = 5$. Ten sam diagram wykorzystuje się w transformacji prostej, jak i odwrotnej, co ułatwia organizację obliczeń. W rozdziale o jednowymiarowej transformacji Fouriera – DFT wykazano, że nieoptymalizowana realizacja tego zadania wymaga wykonania N^2 zespolonych mnożeń oraz $N \cdot (N-1)$ zespolonych dodawań. W przypadku algorytmu Gooda–Thomasa stosowane są jednowymiarowe DFT mniejszych rozmiarów, co pozwala na redukcję złożoności czasowej algorytmu. W omawianym przypadku (rys. 7) pokazano 3 DFT 5-punktowe oraz 5 DFT 3-punktowych, co oznacza, że należy przeprowadzić $3 \cdot 5^2 + 5 \cdot 3^2 = 120$ zespolonych mnożeń oraz $3 \cdot 2 + 5 \cdot 4 = 26$ zespolonych dodawań. Razem należy wykonać 146 operacji arytmetycznych zamiast $15^2 + 15 \cdot 14 = 435$ operacji, jak ma to miejsce w klasycznej transformacji DFT.

Dla dowolnego $N = N_1 \cdot N_2$, $NWD(N_1, N_2) = 1$ transformacja Gooda–Thomasa wymaga $N_1 \cdot N_2^2 + N_2 \cdot N_1^2$ zespolonych mnożeń oraz $N_1(N_1 - 1) + N_2(N_2 - 1)$ dodawań, co pozwala rozwiązać zadanie efektywniej. Rzeczywisty koszt obliczeń jest nieznacznie większy, gdyż w oszacowaniach nie uwzględniono nakładów związanych z porządkowaniem próbek sygnału wejściowego oraz porządkowaniem widma.

Przykład 5.10. *Niech liczba zebranych próbek sygnału wynosi* $N = 15$. *Liczbę tę można zapisać w postaci iloczynu* $N = N_1 \cdot N_2$.
Niech $N_1 = 3$, $N_2 = 5$. *Są to liczby pierwsze. Niech* $x(n) = n$.
Na podstawie (5.137) *oraz* (5.138) *można wskazać próbkę* $x(n)$, *która staje się elementem tablicy* $A(n_1, n_2)$. *W omawianym przykładzie elementy tablicy* A *wypełniane są według następującej reguły*:

$$A(n_1, n_2) = n = 5n_1 + 3n_2 \pmod{15}, \quad n_1 = 0, ..., 2, \quad n_2 = 0, ..., 4.$$



Rys. 7. Schemat przepływowy transformacji Gooda–Thomasa 15-elementowego wektora danych dla $N = 15$, $N_1 = 3$, $N_2 = 5$

Jeśli dane wejściowe tworzą ciąg $x(n)$, *dla* $n = 0, ..., N-1$, *to tablica* A *wypełniona zostanie wartościami próbek w sposób następujący*:

$x(0)$	$x(3)$	$x(6)$	$x(9)$	$x(12)$
$x(5)$	$x(8)$	$x(11)$	$x(14)$	$x(2)$
$x(10)$	$x(13)$	$x(1)$	$x(4)$	$x(7)$

Niech dane wyjściowe tworzą ciąg $s(k)$, *dla* $k = 0, ..., N-1$, *i są przechowywane w tablicy* B. *Na podstawie* (5.139) *otrzymujemy zależność ustalającą porządek zapisywania elementów ciągu* $s(k)$ *w tablicy*:

$$B(k_1, k_2) = k = 10k_1 + 6k_2 \pmod{15}, \quad k_1 = 0, ...2, \quad k_2 = 0, ..., 3,$$

co oznacza, że dane będą umieszczone w tablicy B w następującej kolejności:

$k(0)$	$k(6)$	$k(12)$	$k(3)$	$k(9)$
$k(10)$	$k(1)$	$k(7)$	$k(13)$	$k(4)$
$k(5)$	$k(11)$	$k(2)$	$k(8)$	$k(14)$

.

Poniżej przedstawiono program realizujący transformację Gooda–Thomasa.

Program 5.14. *Prosta transformacja Gooda–Thomasa dla przypadku, gdy ciąg danych wejściowych składa się z* 15 *elementów. Liczbę* $N = 15$ *należy zapisać w formie* $N = N_1 \cdot N_2$. *Przyjęto arbitralnie wartości* $N_1 = 3$, $N_2 = 5$, *tak samo jak w przykładzie* (5.10). *Uwzględniając połączenia diagramu przepływowego na rys.* 7, *otrzymujemy szybką wersję algorytmu.*

```
X=[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14];  % Wektor danych
% Parametry dekompozycji danych wejsciowych
N1=3; N2=5; N=N1*N2;
A=zeros(N1,N2);                      % Przygotowanie tablicy A
% Wypelnienie tablicy A elementami wektora danych
for n1=0:N2-1
    for n2=0:N2-1
        n=mod(n1*N2+n2*N1,N);      A(n1+1,n2+1)=X(n+1);
    end
end
F=zeros(N2);
% Fourierowska macierz transformacji N2xN2
for k=1:N2
    for w=1:N2
        F(w,k)=exp(-j*2*pi*(w-1)*(k-1)/N2);
    end
end
C=zeros(N1,N2);
```

```
% N2-punktowe DFT. Liczba DFT wynosi N1
for i=1:N1
    C(i,:)=A(i,:)*F;
end
F=zeros(N1);
% Fourierowska macierz transformacji N1xN1
for k=1:N1
    for w=1:N1
        F(w,k)=exp(-j*2*pi*(w-1)*(k-1)/N1);
     end
end
Q=zeros(N1,N2);
% N1-punktowe DFT. Liczba DFT wynosi N2
for i=1:N2
    Q(:,i)=C(:,i).'*F
    B=zeros(1,N);                  % Przygotowanie tablicy B
    % Uporzadkowanie wspolczynnikow widmowych
    % w porzadku leksykograficznym
    for k1=0:N1-1
        for k2=0:N2-1
            k=mod(k1*10+k2*6,N);
            B(k+1)=Q(k1+1,k2+1);
        end
    end
end
```

Program może być stosowany prawie bez zmian dla przeprowadzenia transformacji odwrotnej. Jedyną zmianą jest wtedy zamiana znaku $-$ na $+$ w funkcji wykładniczej i uwzględnienie czynników $1/N_2$ oraz $1/N_1$ w N_2 i N_1 punktowych transformacjach DFT, które stają się transformacjami odwrotnymi (IDFT). Należy również przypomnieć, że algorytm Gooda–Thomasa można programować na wiele różnych, równoważnych sposobów, uzyskując tę samą złożoność.

5.7. Funkcje i transformacja Vilenkina–Chrestensona

Funkcje Vilenkina–Chrestensona stosowane są w analizie p-wartościowych funkcji, w szczególności funkcji boolowskich. Z kolei transformacja Vilenkina–Chrestensona ma wiele zalet dyskretnej transformaty Fouriera i znajduje zastosowanie w rozwiązywaniu problemów kodowania informacji [15, 33]. Niektóre przekształcone formy tych funkcji tworzą inne funkcje bazowe, o czym będzie mowa w niniejszym rozdziale.

5.7.1. Funkcje Vilenkina–Chrestensona

Definicja 5.19. *Funkcje Vilenkina–Chrestensona* $ch^{(p)}(0,x), ch^{(p)}(1,x), ..., ch^{(p)}(p^m-1,x)$ *tworzone są w sposób następujący*:

$$ch^{(p)}(w,x) = \exp\left(\frac{2\pi}{p} j \sum_{s=0}^{m-1} w_s x_s\right), \tag{5.142}$$

gdzie: $x, w \in \{0,1,...,p^m-1\}$, $p \geq 2$. *Liczby* m, $p \in \boldsymbol{N}_+$ *oraz*:

$$w = \textstyle\sum_{s=0}^{m-1} w_s p^s, \qquad x = \sum_{s=0}^{m-1} x_s p^s,$$

w_s,$x_s \in \{0,1,...,p-1\}$.

Funkcje podane w Definicji 5.19 tworzą zbiór ortogonalnych funkcji bazowych, są więc parami ortogonalne:

$$< vc^{(p)}(w,x), vc^{(p)}(t,x) >= \sum_{x=0}^{p^m-1} vc^{(p)}(w,x) \cdot vc^{(p)}(t,x) = \begin{cases} p^m & \text{dla } w = t \\ 0 & \text{dla } w \neq t \end{cases} \Leftrightarrow$$
$$||vc^{(p)}(w,x)||^2 = p^m, \quad x \in \{0,1,...,p^m-1\}. \tag{5.143}$$

Z zależności (5.143) wynika, że funkcje Chrestensona nie są unormowane.

Wygenerowane według Definicji 5.19 wartości funkcji można zapisać w postaci macierzy $\boldsymbol{C}_{p^m}$ o wymiarach $p^m \times p^m$. Wiersze tej macierzy utożsamić można z odpowiednią dyskretną funkcją Vilenkina–Chrestensona. Budowa macierzy $\boldsymbol{C}_3$, dla parametrów $p = 3$, $m = 1$, oraz macierzy $\boldsymbol{C}_9$, dla parametrów $p = 3$, $m = 2$, zostały przedstawione poniżej:

$$\boldsymbol{C}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^1 & a^2 \\ 1 & a^2 & a^1 \end{bmatrix}, \quad \boldsymbol{C}_9 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & a^1 & a^2 & 1 & a^1 & a^2 & 1 & a^1 & a^2 \\ 1 & a^2 & a^1 & 1 & a^2 & a^1 & 1 & a^2 & a^1 \\ 1 & 1 & 1 & a^1 & a^1 & a^1 & a^2 & a^2 & a^2 \\ 1 & a^1 & a^2 & a^1 & a^2 & 1 & a^2 & 1 & a^1 \\ 1 & a^2 & a^1 & a^1 & 1 & a^2 & a^2 & a_1 & 1 \\ 1 & 1 & 1 & a^2 & a^2 & a^2 & a^1 & a^1 & a^1 \\ 1 & a^1 & a^2 & a^2 & 1 & a^1 & a^1 & a^2 & 1 \\ 1 & a^2 & a^1 & a^2 & a^1 & 1 & a^1 & 1 & a^2 \end{bmatrix}, \tag{5.144}$$

gdzie: $a = e^{\frac{2\pi}{3}j} = a^1$, $a^2 = e^{\frac{4\pi}{3}j}$.

Macierze Vilenkina–Chrestensona $\boldsymbol{C}_{p^m}$ można również konstruować w sposób rekurencyjny. Niech macierz $\boldsymbol{P}_p$ ma następującą budowę:

$$\boldsymbol{P}_p = \begin{bmatrix} a^0 & a^0 & a^0 & \cdots & a^0 \\ a^0 & a^1 & a^{1\cdot 2} & \cdots & a^{(p-1)\cdot 1} \\ a^0 & a^2 & a^{2\cdot 2} & \cdots & a^{(p-1)\cdot 2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a^0 & a^{(p-1)} & a^{(p-1)\cdot 2} & \cdots & a^{(p-1)\cdot(p-1)} \end{bmatrix}, \tag{5.145}$$

gdzie $a = e^{\frac{2\pi}{p}j}$, a $p \geq 2$.

W zależności od p elementy macierzy $\boldsymbol{P}_p$ przyjmują wartości z zakresu: $a^q = e^{\frac{2q\pi}{p}j}$, dla $q = 0, 1, ..., p-1$. Na przykład dla $p = 4$ otrzymujemy: $a^0 = 1$, $a^1 = e^{\frac{2\pi}{4}j} = j$, $a^2 = e^{\frac{4\pi}{4}j} = -1 = j^2$, $a^3 = e^{\frac{6\pi}{4}j} = -j = j^3$.

Uwzględniając powyższe wywody, macierz Vilenkina–Chrestensona $\boldsymbol{C}_{p^m}$ budowana jest w następujący sposób:

$$\boldsymbol{C}_{p^m} = \bigotimes_{i=1}^{m} \boldsymbol{P}_p = \underbrace{\boldsymbol{P}_p \otimes ... \otimes \boldsymbol{P}_p}_{m-krotnie}. \tag{5.146}$$

Przykład 5.11. *Zbudować macierz* $\boldsymbol{C}_{p^m}$ *w sposób rekurencyjny, na podstawie wzorów* (5.145) *oraz* (5.146)*, dla* $p = 3$ *oraz* $m = 2$*. Dla zadanych parametrów można utworzyć macierz* $\boldsymbol{P}_3$:

$$\boldsymbol{P}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^1 & a^2 \\ 1 & a^2 & a^4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^1 & a^2 \\ 1 & a^2 & a^1 \end{bmatrix},$$

gdyż $a^4 = \left(e^{\frac{2\pi}{3}j}\right)^4 = \left(e^{\frac{2\pi}{3}j}\right) = a^1$*. Ze wzoru* (5.146) *otrzymujemy macierz, której wiersze są dyskretnymi funkcjami Vilenkina–Chrestensona:*

$$\boldsymbol{C}_9 = \boldsymbol{P}_3 \otimes \boldsymbol{P}_3 = \begin{bmatrix} \boldsymbol{P}_3 & \boldsymbol{P}_3 & \boldsymbol{P}_3 \\ \boldsymbol{P}_3 & a^1 \cdot \boldsymbol{P}_3 & a^2 \cdot \boldsymbol{P}_3 \\ \boldsymbol{P}_3 & a^2 \cdot \boldsymbol{P}_3 & a^1 \cdot \boldsymbol{P}_3 \end{bmatrix}.$$

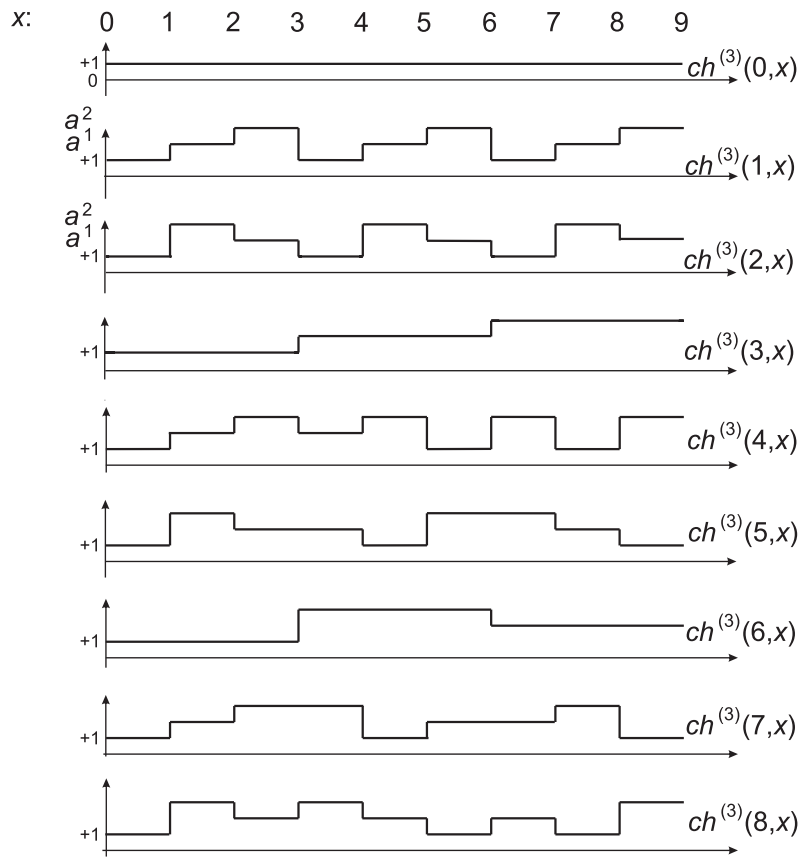
Można zauważyć, że macierz blokowa $\boldsymbol{C}_9$ *jest tylko inaczej zapisaną macierzą* (5.144).

Funkcje Vilenkina–Chrestensona mają również reprezentację graficzną (rys. 8) [15]. Jest to reprezentacja umowna, pokazująca charakter zmian przebiegu tych funkcji. Dla $p = 3$ oraz $m = 2$ przedział określoności został podzielony na dziewięć jednakowych podprzedziałów. Omawiane funkcje mają przebieg funkcji odcinkowo-stałych w każdym z wydzielonych podprzedziałów $\left[\frac{n}{p^m}, \frac{n+1}{p^m}\right)$, dla $n = 0, 1, ..., p^m - 1$. Macierz funkcji Vilenkina–Chrestensona można wygenerować za pomocą programu zapisanego w składni Matlaba, wykorzystując dla zrealizowania tego celu podstawową Definicję 5.19 tych funkcji. Macierz taką można następnie skonfrontować z odpowiednimi wykresami funkcji.

Program 5.15. *Program w języku Matlab pozwalający na utworzenie macierzy* $\boldsymbol{C}_{p^m}$ *o rozmiarach* $p^m \times p^m$*.*

```
p=3;          % Liczba okreslajaca rozmiary macierzy C_{p^{m}}
m=2;          % Liczba okreslajaca rozmiary macierzy C_{p^{m}}
ex=((2*pi)/p)*i;                    % Zmienna pomocnicza
```

```
% Zostanie zbudowana macierz VH o rozmiarach p^{m} x p^{m}
VH = zeros(p^m,p^m);
for w=0:(p^m)-1
    for z=0:(p^m)-1
        c=0;                      % Zmienna pomocnicza
         % Zamiana liczby w na znaki Ws w syst. o podst. p
        Ws=dec2base(w,p,m);
         % Zamiana liczby z na znaki Zs w syst. o podst. p
        Zs=dec2base(z,p,m);
         % m - liczba pozycji w reprezentacji o podst. p
        for s=1:m
            aa=str2num(Ws(s));   % Przeksztalcenie znaku
                                   % Ws(s) w liczbe
            bb=str2num(Zs(s));   % Przeksztalcenie znaku
                                   % Zs(s) w liczbe
            cc=aa*bb;            % Mnozenie liczb na pozycjach
                                   % znakow Ws oraz Zs
            c=c+cc;              % Przyrostowe sumowanie liczb
         end
         VH(w+1,z+1)=exp(ex*c);    % Element macierzy
    end                            % Vilenkina-Chrestensona
end
VH                                 % Kompletna macierz VH
```



Rys. 8. Umowna reprezentacja graficzna funkcji Vilenkina–Chrestensona dla $p = 3$ oraz $m = 2$. UWAGA! Oś rzędnych nie jest skalowana

Definicja 5.20. *Jednowymiarową transformacją Vilenkina–Chrestensona sygnału $x(n)$ określonego dla $n = 0, 1, ..., p^m - 1$, nazywamy odwzorowanie, w wyniku którego otrzymujemy następującą transformatę:*

$$s(k) = \alpha \sum_{n=0}^{p^m-1} x(n) \cdot vc^{(p)}(n,k), \qquad k = 0, 1, ..., p^m - 1, \tag{5.147}$$

gdzie: $vc^{(p)}(n,k)$ jest n-tą funkcją reprezentowaną przez k-ty wektor wierszowy macierzy $\boldsymbol{C}_{p^m}$.

Definicja 5.21. *Jednowymiarową, odwrotną dyskretną transformacją Vilenkina–Chrestensona jest odwzorowanie, które na podstawie widma sygnału* $s(k)$, $k = 0, ..., p^m - 1$ *odtwarza próbki sygnału pierwotnego* $x(n)$:

$$x(n) = \beta \sum_{k=0}^{p^m-1} s(k) \cdot vc^{(p)*}(n,k), \qquad n = 0, 1, ..., p^m - 1, \tag{5.148}$$

gdzie:
funkcja $vc^{(p)*}(n,k)$ *jest sprzężona do funkcji Vilenkina–Chrestensona* $vc^{(p)}(n,k)$.

Dla macierzy $\boldsymbol{C}_{p^m}$ zachodzi więc następujący związek:

$$(\boldsymbol{C}_{p^m})^{-1} = \frac{1}{p^m}(\boldsymbol{C}_{p^m})^*. \tag{5.149}$$

Uwzględniając znaną już własność (5.149), jednowymiarową transformację Vilenkina–Chrestensona przedstawić można w zapisie macierzowym:

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{x} \cdot \boldsymbol{C}_{p^m}, \tag{5.150}$$

$$\boldsymbol{x} = \boldsymbol{s} \cdot (\boldsymbol{C}_{p^m})^{-1} = \beta \cdot \boldsymbol{s} \cdot (\boldsymbol{C}_{p^m})^*. \tag{5.151}$$

Transformacje (5.150) oraz (5.151) będą wzajemnie odwrotne, jeśli spełniony będzie warunek:

$$\alpha \cdot \beta = \frac{1}{p^m}. \tag{5.152}$$

Przykład 5.12. *Wyznaczyć współczynniki widmowe transformacji Vilenkina–Chrestensona wektora* $\boldsymbol{x} = [0, 1, 2]$, *korzystając z wzoru (5.150). Zgodnie z przeprowadzonymi już wywodami, mamy:* $p = 3$ *oraz* $m = 1$. *Przyjmując* $\alpha = 1$, $\beta = 1/3$, *otrzymujemy*:

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{x} \cdot \boldsymbol{C}_3 = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & a^1 & a^2 \\ 1 & a^2 & a^1 \end{bmatrix} =$$

$$= \begin{bmatrix} 3 & (a^1 + 2a^2) & (a^2 + 2a^1) \end{bmatrix} = \Big[3 \quad (-1.5 - 0.866j) \quad (-1.5 + 0.866j)\Big],$$

gdzie $a = e^{\frac{2\pi}{3}j}$.

Odwracanie macierzy $\boldsymbol{C}_3$ *można sprowadzić do wyznaczenia macierzy sprzężonej* $(\boldsymbol{C}_{p^m})^*$:

$$(\boldsymbol{C}_{p^m})^* = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \overline{a^1} & \overline{a^2} \\ 1 & \overline{a^2} & \overline{a^1} \end{bmatrix}.$$

Znając widmo, można odtworzyć oryginalne wartości próbek:

$$\boldsymbol{x} = \beta \cdot \boldsymbol{s} \cdot \boldsymbol{W}^* = \frac{1}{3} \cdot \begin{bmatrix} 3 & (-1.5 - 0.866j) & (-1.5 + 0.866j) \end{bmatrix} \cdot$$
$$\cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & \overline{a^1} & \overline{a^2} \\ 1 & \overline{a^2} & \overline{a^1} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}.$$

5.7.2. Dyskretna szybka transformacja Vilenkina–Chrestensona

Obliczenie widma Vilenkina-Chrestensona metodą bezpośrednią jest nieefektywne, chyba że liczba próbek sygnału spełnia warunek $N = \log_2 q^m$, czyli jest potęgą liczby 2. W takim przypadku można zastosować opisywany już algorytm szybkiej transformacji Fouriera. Jeśli liczba N nie jest potęgą liczby 2, to należy zastosować tzw. szybki algorytm Cooleya–Tukeya, dla $N = L \cdot M$, gdzie M oraz L są liczbami naturalnymi. Innym rozwiązaniem jest zastosowanie algorytmu Gooda–Thomasa, w którym N może być przedstawione jako iloczyn liczb względnie pierwszych – ich jedynym wspólnym podzielnikiem jest 1. Konsekwencją innej reprezentacji liczby N jest również inne uporządkowanie danych wejściowych, dla których należy wyznaczyć transformatę.

Szybką transformację Vilenkina–Chrestensona można zrealizować programowo, wykorzystując na przykład do tego celu zmodyfikowany Program 5.14, realizujący szybką transformację Gooda–Thomasa oraz Program 5.15. W takim przypadku dodatkowymi parametrami nowego programu będą liczby p oraz m, a zamiast macierzy fourierowskiej wykorzystana będzie macierz uzyskiwana za pomocą programu (Program 5.15).

Funkcje Vilenkina–Chrestensona mogą również przybierać o wiele prostszą formę, w której nie występują wartości urojone. Analizując wzór (5.142), można zauważyć, że dla $p = 2$ wartość $e^{\pi jq} = \pm 1$ dla dowolnej wartości q. Dla tych założeń funkcje Vilenkina–Chrestensona przekształcają się do funkcji Walsha. Otrzymane w ten sposób funkcje Walsha są uporządkowane według tzw. porządku Hadamarda, co objaśnione zostanie dokładniej w następnym podrozdziale.

5.8. Funkcje i transformacje Walsha

Stosowanie funkcji zespolonych w transformacji Fouriera lub transformacji Vilenkina–Chrestensona powoduje, że również transformata zawiera elementy zespolone, co często niepotrzebnie komplikuje obliczenia. W praktyce bardzo często mamy do czynienia z sygnałami, które przyjmują wyłącznie wartości rzeczywiste lub nawet naturalne, np. w komputerowych obrazach bitmapowych czy funkcjach boolowskich, tak więc istnieje oczywista potrzeba posługiwania się transformacjami, gdzie bazami są funkcje innego typu. W przypadku takich transformacji funkcjami bazowymi mogą być funkcje Walsha lub Haara.

Funkcje Walsha są bipolarnymi falami prostokątnymi. Są to funkcje binarne, odcinkami stałe, przyjmujące tylko jedną z dwóch wartości: -1 lub $+1$. Funkcje te mają rozbudowany, mało czytelny opis definicyjny, ale wyjątkowo prostą reprezentację graficzną, a w konsekwencji także prostą interpretację. Funkcje te są chętnie wykorzystywane z uwagi na fakt, że mają reprezentację zarówno w przestrzeniach funkcyjnych, jak i przestrzeniach wektorowych. Opis tych funkcji w przestrzeni funkcyjnej $L^2[0,1)$ podał w 1923 r. J.L. Walsh. Bazą do ich utworzenia były znane wcześniej funkcje Rademachera [4,17]. W tej przestrzeni funkcje Walsha tworzą zupełny zbiór funkcji ortonormalnych i wykazują wiele analogii z funkcjami trygonometrycznymi *sin* i *cos*. W literaturze można znaleźć wiele, często równoważnych, sposobów generowania tych funkcji [11,13,17,28,34,40]. Funkcje Walsha oznaczane jako $Wal(x,t)$ opisane są dwoma wskaźnikami. Wskaźnik x określa numer (rząd) funkcji, a wskaźnik t – bieżącą wartość funkcji w przedziale określoności.

Przedziały określoności funkcji Walsha mogą dotyczyć dowolnego, równego okresowi, przedziału. Spotyka się opisy definiujące funkcje Walsha w przedziale $-\frac{1}{2} \leq t < \frac{1}{2}$ [14] lub w przedziale $0 \leq t < 1$ [17]. Poza zdefiniowanym przedziałem określoności, wartość każdej funkcji Walsha wynosi 0. Wyznaczanie kolejnych funkcji Walsha przebiega iteracyjnie. Oryginalny opis tych funkcji opiera się na harmonicznych funkcjach prostokątnych. Wydaje się skomplikowany, dlatego nie jest stosowany w operacjach dyskretnych – zostanie tutaj jednak przytoczony, gdyż jest niezbędny w rozważaniach dotyczących przestrzeni funkcyjnych. Zawiły opis funkcji Walsha można zastąpić równoważnym, prostszym zapisem.

Definicja 5.22. *Niech w przestrzeni* $L^2[0,1)$ *istnieje podprzestrzeń funkcyjna* $\boldsymbol{V}$ *taka, że* $\dim \boldsymbol{V} = N$. *W podprzestrzeni* $\boldsymbol{V}$ *jest zdefiniowana funkcja Walsha* $Wal(0,t)$ *taka, że*:

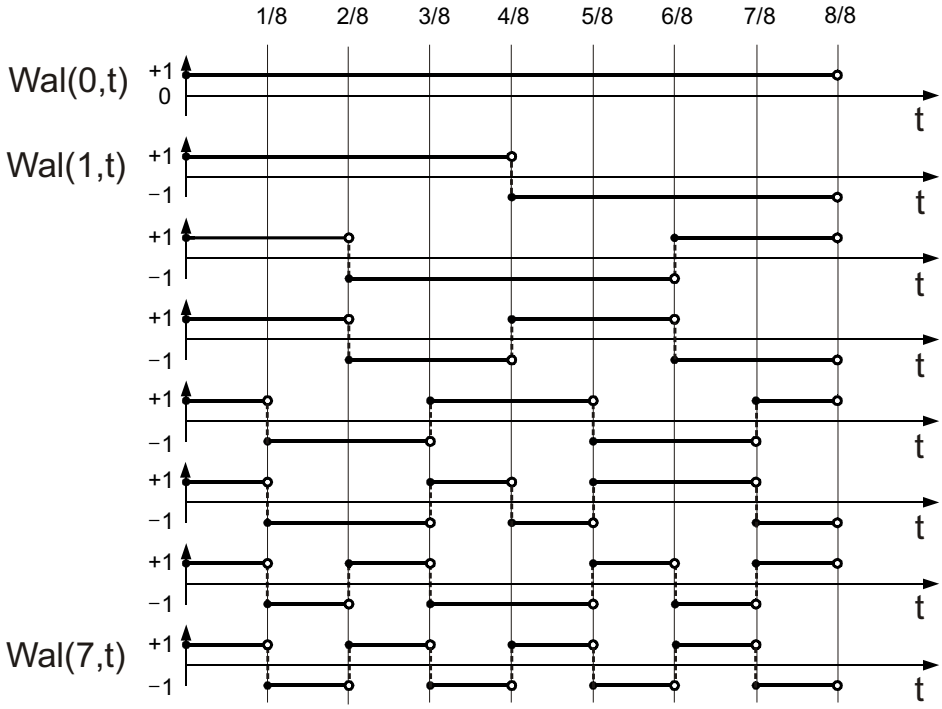
$$Wal(0,t) = \begin{cases} +1 & \text{dla} \quad 0 \leq t < 1 \\ 0 & \text{dla} \quad t < 0 \ i \ t \geq 1 \end{cases}, \tag{5.153}$$

którą można przedłużać okresowo z okresem 1 *na całą oś liczbową* t.
Funkcje Walsha wyższych rzędów można wyznaczyć iteracyjnie na podstawie bazowej funkcji (5.153) *w sposób następujący*:

$$Wal(2j+p,t) = Wal(j,2t) + (-1)^{j+p} Wal(j,2t-1), \tag{5.154}$$

gdzie: $p = 0,1$, $j = 0,1,...,N/2-1$, $\max(2j+p) = N-1$ *oraz zakłada się, że* N *jest potęgą dwójki.*

Wykres pierwszych ośmiu funkcji Walsha wyznaczonych według formuły (5.154) został przedstawiony na rys. 9, gdzie przedział określoności $[0,1)$ został podzielony dodatkowo na osiem jednakowych podprzedziałów $\left[\frac{n}{N}, \frac{n+1}{N}\right)$ dla $N = 8$, $n = 1,...,7$. Przebieg tych funkcji jest zgodny z definicją podaną przez J.L. Walsha. Funkcje Walsha posiadają skończoną liczbę punktów nieciągłości wewnątrz wybranego przedziału określoności. Wartości funkcji w punktach nieciągłości wewnątrz tego przedziału mogą być dowolne. W szczególności mogą to być wartości równe $\frac{1}{2}$ [34] lub też wartości równe średniej arytmetycznej granicy lewo- i prawostronnej w tych punktach [17].



Rys. 9. Wykres pierwszych ośmiu ($N = 8$) funkcji Walsha określonych na przedziale $[0, 1)$

Można łatwo zauważyć, że funkcje Walsha są w całym przebiegu funkcjami odcinkowo-stałymi i przyjmują stałą wartość $+1$ lub -1 w każdym z podprzedziałów $\left[\frac{n}{N}, \frac{n+1}{N}\right)$, gdzie $n = 0, 1, ..., N-1$. Zaprezentowane na rys. 9 funkcje są uporządkowane w tzw. porządku Walsha, co oznacza, że kolejne numery funkcji są kojarzone z liczbą przejść przez zero (liczbą przecięć osi czasu) każdej z nich. Spotyka się również inne uporządkowania tych funkcji. Mogą one występować na przykład w porządku Kaczmarza (Paleya), Rademachera lub Hadamarda [9, 28].

Każde z wymienionych uporządkowań jest równoprawne i zawsze zawiera zbiór tych samych funkcji Walsha. Sposób porządkowania funkcji Walsha, chociaż da się wyrazić formułami matematycznymi, nie ma większego znaczenia i stosuje się opisy funkcji, które są łatwe w implementacjach programowych.

Funkcje Walsha charakteryzuje ciekawa własność – są multiplikatywne i symetryczne, co oznacza, że iloczyn dowolnych funkcji Walsha jest także funkcją Walsha:

$$Wal(a,m)\cdot Wal(b,m) = Wal(a\oplus b,m), \tag{5.155}$$

$$Wal(a,m)\cdot Wal(a,n) = Wal(a,m\oplus n), \tag{5.156}$$

gdzie $\oplus$ jest sumą modulo 2 wyrażonych binarnie numerów funkcji, co oznacza, że mnożenie dwóch funkcji Walsha polega na arytmetycznym sumowaniu bez przeniesienia zapisanych binarnie numerów tych funkcji. Na przykład $Wal(3,t)\cdot Wal(5,t) = Wal(011\otimes 101,t) = Wal(110,t) = Wal(6,t)$.
Funkcje Walsha są parami ortogonalne:

$$\langle Wal(i,t), Wal(j,t)\rangle = \int_0^1 Wal(i,t)\cdot Wal(j,t)dt = \begin{cases} 1 & dla \quad i=j \\ 0 & dla \quad i\neq j \end{cases} \tag{5.157}$$

oraz unormowane:

$$\|Wal(i,t)\| = \left[\int_0^1 Wal^2(i,t)dt\right]^{1/2} = 1. \tag{5.158}$$

Przykład 5.13. *Obserwując przebieg funkcji Walsha pokazanych na rys.* 9*, wyznaczyć normę oraz iloczyn skalarny dowolnych dwóch funkcji Walsha*:

$$\langle Wal(2,t), Wal(2,t)\rangle = \int_0^{1/4}(+1)\cdot(+1)dt + \int_{1/4}^{3/4}(-1)\cdot(-1)dt + \int_{3/4}^{1}(+1)\cdot(+1)dt = \tfrac{1}{4}+\tfrac{2}{4}+\tfrac{1}{4} = 1,$$

$$\langle Wal(1,t), Wal(2,t)\rangle = \int_0^{1/4} (+1)\cdot(+1)dt + \int_{1/4}^{2/4} (+1)\cdot(-1)dt + \int_{2/4}^{3/4} (-1)\cdot(-1)dt +$$
$$+ \int_{3/4}^{1} (-1)\cdot(+1)dt = \tfrac{1}{4} - \tfrac{1}{4} + \tfrac{1}{4} - \tfrac{1}{4} = 0,$$

oraz

$$\|Wal(2,t)\| = [\langle Wal(2,t), Wal(2,t)\rangle]^{1/2} = 1.$$

Zaprezentowane na rys. 9 przebiegi funkcji można generować za pomocą programu komputerowego. Poniżej przedstawiono przykład programu napisanego w języku Matlab, który posłużył do wygenerowania przebiegów takich funkcji. Zainteresowany Czytelnik może ten program wykorzystywać w samodzielnych eksperymentach.

Program 5.16. *Program w języku Matlab, za pomocą którego można, zgodnie z Definicją 5.22, obrazować przebieg funkcji w porządku Walsha. Przebieg tych funkcji będzie zgodny z wynikami przedstawionymi na rys.* 9.

```
N=8;                      % Wybor liczby funkcji. Wymiar przestrzeni
N=pow2(floor(log2(N)));   % Liczba N musi byc potega 2!!
for i=0:N-1               % Wykreslanie funkcji bazowych
    subplot(N,1,i+1);
    Walsh_graph(i);
end
%% CIALO FUNKCJI %%
function Walsh_graph(N)          % Synteza funkcji Walsha
for i = 1:1500
    x(i) = ((i./1000.0)-0.25);
    y(i) = Wal(N, x(i));
end
k = plot(x,y);
set(k,'LineWidth',2.5);
```

```
xlim([-0 1.]);
ylim([-1.5 1.5]);
grid on
end
function W=Wal(N,t)
if(N==0)
   if(t<0)
      W=0; return;
      elseif (t>=1)
      W=0; return;
   else
      W=1; return;
   end
else
   if (mod(N, 2.0)~=0)
      p=1;
   else
      p=0;
   end
   j = fix(N./2.0);
   W=(Wal(j,2*t)+(-1).^(j+p)*Wal(j,2*t-1));
   return;
end
end
```

5.8.1. Dyskretne funkcje Walsha

Dyskretne funkcje Walsha, w odróżnieniu od funkcji ciągłych, oznaczane będą małą literą $wal(x,t)$. Dyskretne funkcje Walsha otrzymujemy w rezultacie dyskretyzacji ciągłych funkcji Walsha, zawierających skończoną liczbę punktów nieciągłości w przedziale $[0,1)$. Dyskretyzacja przeprowadzana jest w ten sposób, że cały przedział określoności każdej funkcji Walsha dzielony jest na

N elementarnych podprzedziałów i w środku każdego podprzedziału $\left[\frac{n}{N}, \frac{n+1}{N}\right)$, $n = 0, ..., N-1$, $N = 2^m$ pobierana jest wartość wybranej próbki funkcji Walsha. Zbiór utworzonych w ten sposób dyskretnych punktów wygodnie jest przedstawić w postaci macierzy. Okazuje się, że może to być tzw. macierz Hadamarda. Poszczególne wiersze macierzy mają tzw. porządek Hadamarda. Każdy wiersz macierzy Hadamarda można traktować jako N elementowy wektor – dyskretną funkcję Walsha. Wektory te tworzą bazę przestrzeni liniowej.

Definicja 5.23. *Macierzą Hadamarda stopnia* 2^m, $m = 0, 1, ...$ *nazywamy kwadratową macierz* $\boldsymbol{H}_m$ *o rozmiarach* $2^m \times 2^m$ *o elementach równych* ± 1, *spełniającą warunek*:

$$\boldsymbol{H}_m \cdot \boldsymbol{H}_m^T = m\boldsymbol{I}_m \ , \tag{5.159}$$

gdzie $\boldsymbol{I}_m$ *jest macierzą jednostkową o wymiarze* $2^m \times 2^m$.

Macierze Hadamarda stopnia 2^m mogą być generowane jako wynik kronekerowskiego iloczynu macierzy stopnia 2^{m-1}. Macierze Hadamarda mogą więc być budowane w sposób rekurencyjny:

$$\boldsymbol{H}_m = \bigotimes_{i=1}^{m} \boldsymbol{H}_1 = \boldsymbol{H}_1 \otimes \ \boldsymbol{H}_{m-1} = \underbrace{\boldsymbol{H}_1 \otimes ... \otimes \boldsymbol{H}_1}_{m-krotnie}, \tag{5.160}$$

gdzie $\boldsymbol{H}_0 = [1]$, $\boldsymbol{H}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ są elementarnymi macierzami Hadamarda.

Wzór (5.160) można również zapisać inaczej:

$$\boldsymbol{H}_m = \begin{bmatrix} \boldsymbol{H}_{m-1} & \boldsymbol{H}_{m-1} \\ \boldsymbol{H}_{m-1} & -\boldsymbol{H}_{m-1} \end{bmatrix}, \tag{5.161}$$

gdzie $\boldsymbol{H}_m$ jest macierzą kwadratową o rozmiarze $2^m \times 2^m$.

Okazuje się, że każdy wiersz lub kolumnę macierzy Hadamarda można traktować jak wektor utożsamiany z odpowiednią dyskretną funkcją Walsha $wal(w, t)$, $w, t = 0, ..., 2^m - 1$.

Na przykład macierz $\boldsymbol{H}_3$ można rekurencyjnie wyznaczyć ze wzoru (5.161):

$$\boldsymbol{H}_3 = \boldsymbol{H}_1 \otimes \boldsymbol{H}_1 \otimes \boldsymbol{H}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}. \quad (5.162)$$

Można zauważyć, że macierz Hadamarda zawiera $2^m(2^m-1)/2$ elementów o wartości -1 oraz $2^m(2^m+1)/2$ elementów o wartościach $+1$, a w dowolnych dwóch sąsiadujących ze sobą wierszach (kolumnach) macierzy $\boldsymbol{H}_m$ połowa odpowiadających sobie elementów ma taki sam znak, a połowa zaś przeciwny. Elementy h_{kj} macierzy $\boldsymbol{H}_m$ można wyznaczyć także z zależności [28]:

$$h_{kj} = (-1)^{\overset{m-1}{\underset{b=0}{\oplus}} k_b j_b}, \quad (5.163)$$

gdzie $k_b, j_b = 0, 1, ..., 2^m - 1$ są binarnymi reprezentacjami liczb naturalnych k oraz j, wskazującymi bieżący wiersz i kolumnę macierzy $\boldsymbol{H}_m$.

Dyskretne funkcje Walsha opisane macierzą $\boldsymbol{H}_m$ są parami ortogonalne oraz nieunormowane:

$$< wal(i,t), wal(j,t) > \; = \sum_{t=0}^{2^m-1} wal(i,t) \cdot wal(j,t) = \begin{cases} 2^m & dla \quad i = j \\ 0 & dla \quad i \neq j \end{cases} \quad (5.164)$$

oraz

$$\|wal(i,t)\| = \sqrt{2^m}. \quad (5.165)$$

Wiersze macierzy $\boldsymbol{H}_m$ mogą być unormowane do jedności:

$$\widetilde{\boldsymbol{H}}_m = \frac{1}{\sqrt{N}} \begin{bmatrix} \boldsymbol{H}_{m-1} & \boldsymbol{H}_{m-1} \\ \boldsymbol{H}_{m-1} & -\boldsymbol{H}_{m-1} \end{bmatrix}. \tag{5.166}$$

Operacja normowania jest obliczeniowo kosztowna i ze względów praktycznych często nie jest wykonywana.
Z własności macierzy Hadamarda wynika ponadto, że zachodzą dla niej proste zależności:

$$\boldsymbol{H}_m = \boldsymbol{H}_m^T, \text{ oraz } \quad \boldsymbol{H}_m^{-1} = \frac{1}{2^m}\boldsymbol{H}_m. \tag{5.167}$$

W programie Matlab macierz Hadamarda o rozmiarach $N \times N$, gdzie $N = 2^m$, można wygenerować poleceniem **hadamard**(**N**). Macierz Hadamarda można również utworzyć w programie Matlab, korzystając ze wzoru (5.163).

Program 5.17. *Program w języku Matlab pozwalający na tworzenie macierzy Hadamarda* $\boldsymbol{H}$ *o wymiarach* $N \times N$. *Macierz budowana jest na podstawie wzoru definicyjnego* (5.163).

```
N=4;                          % Rozmiary macierzy Hadamarda
N=pow2(floor(log2(N)));       % Liczba N musi byc potega 2!!
H=zeros(N,N);
b=0;
for k=1:N
    for j=1:N
        w=bitand(k-1,j-1);    % Iloczyn logiczny dwoch
                              % binarnych wartosci
        f=dec2bin(w);         % Zamiana liczby w na wartosc binarna
        b=sum(f);             % Suma bitow o wartosci 1 binarnego
                              % wektora f
        H(k,j)=(-1)^b;        % Wyznaczenie wartosci elementu
                              % macierzy H
    end
end
```

Wykorzystując możliwości języka Matlab, macierz Hadamarda można również budować, korzystając z polecenia $\mathbf{kron}(\boldsymbol{A}, \boldsymbol{B})$ realizującego kronekerowskie mnożenie macierzy $\boldsymbol{A}$ oraz $\boldsymbol{B}$.

Jeśli elementarną macierzą Hadamarda jest macierz $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, to kolejne, większe macierze otrzymać można następująco:
$\boldsymbol{B} = \mathbf{kron}(\boldsymbol{A}, \boldsymbol{A})$, $\boldsymbol{C} = \mathbf{kron}(\boldsymbol{B}, \boldsymbol{A})$, $\boldsymbol{D} = \mathbf{kron}(\mathbf{C}, \mathbf{A})$ itd., co można wykonywać rekurencyjne aż do osiągnięcia żądanej wielkości macierzy.

5.8.2. Dyskretna jednowymiarowa transformacja Walsha–Hadamarda

Zbiór funkcji Walsha może być porządkowany na wiele sposobów. W poprzednim rozdziale opisano tzw. porządek Walsha, gdzie kolejne numery funkcji odpowiadają wzrastającej liczbie jej przejść przez zero. Dla takiego porządku można zbudować transformację dla dowolnego $N \in \boldsymbol{N}$. Jednym z innych porządków stosowanych w zadaniach dyskretnych jest uporządkowanie Walsha–Hadamarda wynikające z budowy macierzy (5.161).

Ponieważ macierz Hadamarda $\boldsymbol{H}_m$ jest macierzą kwadratową o rozmiarach $2^m \times 2^m$, dyskretna transformacja Walsha–Hadamarda wymaga, aby liczba elementów sygnału dyskretnego była wielokrotnością liczby 2.

Definicja 5.24. *Jednowymiarową dyskretną transformacją Walsha–Hadamarda (ang. Discrete Walsh-Hadamard Transform – DWT) sygnału $x(n)$ określonego w chwilach $n = 0, 1, ..., N-1$, $N = 2^m$ nazywamy odwzorowanie, w wyniku którego otrzymujemy następującą transformatę*:

$$s(k) = \alpha \sum_{n=0}^{N-1} x(n) \cdot wal(n, k), \qquad k = 0, 1, ..., N-1, \tag{5.168}$$

gdzie $wal(n, k)$ jest n-tą dyskretną funkcją Walsha reprezentowaną przez k-ty wektor kolumnowy macierzy Hadamarda $\boldsymbol{H}_m$.

W podobny sposób można zdefiniować proces odwrotny.

Definicja 5.25. *Jednowymiarową dyskretną odwrotną transformacją Walsha–Hadamarda (ang. Inverse Discrete Walsh–Hadamard Transform – IDWHT) jest odwzorowanie, które na podstawie próbek $s(k)$, $k = 0, ..., N-1$ widma sygnału odtwarza próbki sygnału pierwotnego $x(n)$:*

$$x(n) = \beta \sum_{k=0}^{N-1} s(k) \cdot wal(n,k), \qquad n = 0, 1, ..., N-1, \tag{5.169}$$

gdzie $wal(n,k)$ jest n-tą dyskretną funkcją Walsha, reprezentowaną przez n-ty wiersz macierzy Hadamarda $\boldsymbol{H}_m$ – jest to n-ta kolumna macierzy $\boldsymbol{H}_m^T$.

Uwzględniając, że dla macierzy Hadamarda spełnione są własności (5.167), zapis macierzowy jednowymiarowych transformacji Walsha można przedstawić w sposób następujący:

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{x} \cdot \boldsymbol{H}_m, \tag{5.170}$$

$$\boldsymbol{x} = \beta \cdot \boldsymbol{s} \cdot \boldsymbol{H}_m^T = \beta \cdot \boldsymbol{s} \cdot \boldsymbol{H}_m. \tag{5.171}$$

Transformacje (5.170) oraz (5.171) będą wzajemnie odwrotne, jeśli spełniony będzie warunek:

$$\alpha \cdot \beta = \frac{1}{N}. \tag{5.172}$$

Przykład 5.14. *Prosta oraz odwrotna transformacja Walsha–Hadamarda wektora danych $\boldsymbol{x} = [1, 2, -2, 0]$. $N = 4$, $m = \log_2 N = 2$. Przyjmując $\alpha = 1$, $\beta = \frac{1}{4}$, otrzymujemy:*

$$\boldsymbol{s} = \boldsymbol{x} \cdot \boldsymbol{H}_2 = \begin{bmatrix} 1 & 2 & -2 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -3 & 5 & 1 \end{bmatrix},$$

$$\begin{aligned} \boldsymbol{x} &= \frac{1}{4} \cdot \boldsymbol{s} \cdot \boldsymbol{H}_2 = \frac{1}{4} \cdot \begin{bmatrix} 1 & -3 & 5 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} 1 & 2 & -2 & 0 \end{bmatrix}. \end{aligned}$$

Programy prostej oraz odwrotnej transformacji Walsha–Hadamarda mogą być realizowane bezpośrednio na podstawie zależności (5.170) oraz (5.171). Jest to jednak rozwiązanie trywialne, bez znaczenia praktycznego, gdyż jego złożoność obliczeniowa wynosi $O(N^2)$. Transformatę Walsha można jednak zrealizować analogicznie do znanej szybkiej transformacji FFT [40]. Na rys. 3 przedstawiono iteracyjny graf przepływowy prostej, szybkiej transformacji Walsha–Hadamarda. Za pomocą tego grafu można wyznaczyć współczynniki widmowe w porządku Walsha–Hadamarda.

5.8.3. Dyskretna szybka transformacja Walsha–Hadamarda

Właściwości funkcji Walsha, a zwłaszcza ich binarny charakter ułatwiają implementację tej transformacji za pomocą komputera. W praktyce obliczenia oparte są na algorytmie szybkiej transformacji Walsha, wykorzystującym jedynie operacje sumowania i odejmowania. Szybka transformacja Walsha wykonuje się w czasie $O(N \log_2 N)$, podobnie jak w FFT. Złożoność pamięciowa tej transformacji Walsha jest równa liczbie potrzebnych do zapamiętania elementów wektora wejściowego, wynosi więc $O(N)$. Z własności macierzy Hadamarda wynikają omówione już zależności: $\boldsymbol{H}_m = \boldsymbol{H}_m^T$ oraz $\boldsymbol{H}_m^{-1} = \frac{1}{2^m}\boldsymbol{H}_m$, co oznacza, że w prostych i odwrotnych przekształceniach można stosować te same macierze przekształceń, co znacznie upraszcza proces obliczeń. Organizacja szybkiej transformacji Walsha zaprezentowana została na rys. 10 w postaci grafu przepływowego. Dla celów poglądowych na rysunku przedstawiono etapy obliczeń widma Walsha dla przykładowego wektora wejściowego $\boldsymbol{x} = [1, 2, 3, 4, 5, 6, 7, 8]$. Elementy grafu przepływowego mają motylkową, regularną budowę, znaną z szybkiej transformacji Fouriera FFT [28, 40]. W przeciwieństwie do operacji zespolonych wymaganych w transformacji FFT, występują tutaj tylko elementarne, proste w zaprogramowaniu operacje dodawania i odejmowania. Czytając schemat zamieszczony na rys. 10 odwrotnie, tzn. od strony prawej do lewej, otrzymujemy, z dokładnością do czynnika normującego, przepis na odwrotną, szybką transformację Walsha. Prosta i odwrotna transformacja Walsha–Hadamarda mogą być więc opisane tym samym algorytmem.

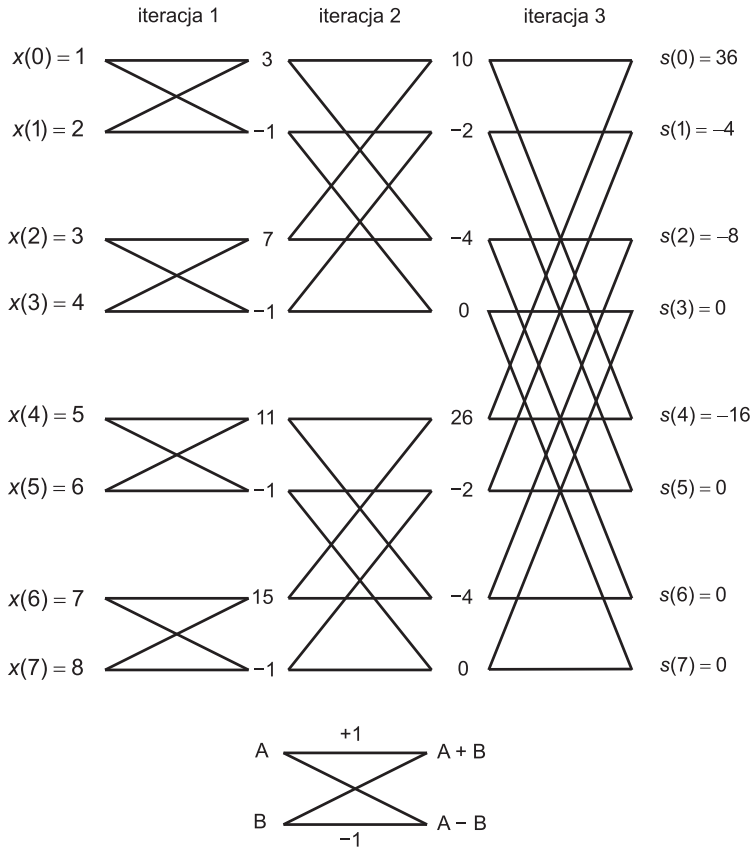
Schemat przepływowy odwrotnej transformacji został natomiast przedstawiony na rys. 11. Ponieważ macierze Hadamarda $\boldsymbol{H}_m$ są symetryczne, a ich odwrotność jest tą samą macierzą z dokładnością do czynnika normującego $\frac{1}{2^m}$, to procedura wyznaczania transformacji odwrotnej może przebiegać według takiego samego schematu jak transformacja prosta.

Poniższe schematy mają strukturę z wyraźnie zaznaczonymi operacjami motylkowymi. Zamiast schematu pokazanego na rys. 11, można zastosować ponownie schemat z rys. 10, uwzględniając tym razem w końcowych obliczeniach stałą normującą. W algorytmach tego typu, zarówno w prostej, jak i odwrotnej transformacji, w każdej iteracji wykonywanych jest N operacji dodawania i odejmowania. Liczba iteracji wynosi $m = \log_2 N$, co zaznaczono na rysunkach. W iteracjach występuje wyraźny podział grupowy, co można zaobserwować na rys. 10. W każdej iteracji liczbę tych grup wyznaczyć można z zależności $r = 2^{\#i-1}$, gdzie $\#i$ jest numerem bieżącej iteracji. Złożoność obliczeniowa takich algorytmów wynosi $O(N \log_2 N)$ i jest taka sama jak w algorytmie FFT. Obliczane w bieżącej iteracji wartości mogą być nadpisywane w pamięci komputera przez wartości wyznaczone w poprzednim kroku, tak więc algorytm nie wymaga rezerwacji dodatkowej pamięci na żadne elementy tymczasowe.

Podobne grafy przepływowe można budować dla innych porządków bazowych funkcji Walsha, rozpinających N wymiarową przestrzeń liniową [9, 40]. Ze względu na właściwości macierzy Hadamarda, dla których spełnionione jest równanie $\boldsymbol{H}_m^{-1} = \frac{1}{2^m}\boldsymbol{H}_m$, schematy grafowe dla prostej i odwrotnej transformacji Walsha można organizować w identyczny sposób, co pokazano za pomocą schematów przepływowych.

Każdy z omówionych wyżej schematów może być równoprawnie zastosowany, a ich programowanie jest bardzo podobne. Poniżej przedstawiono program w postaci procedury języka Matlab (zgodny ze schematem prezentowanym na rys. 3). Ten sam schemat grafowy posłużył do wyznaczenia prostej lub odwrotnej transformacji Walsha. Znane są także inne algorytmy szybkich transformat Walsha, pozwalające uzyskiwać współczynniki w innym porządku, oczywiście zawsze o tych samych wartościach.

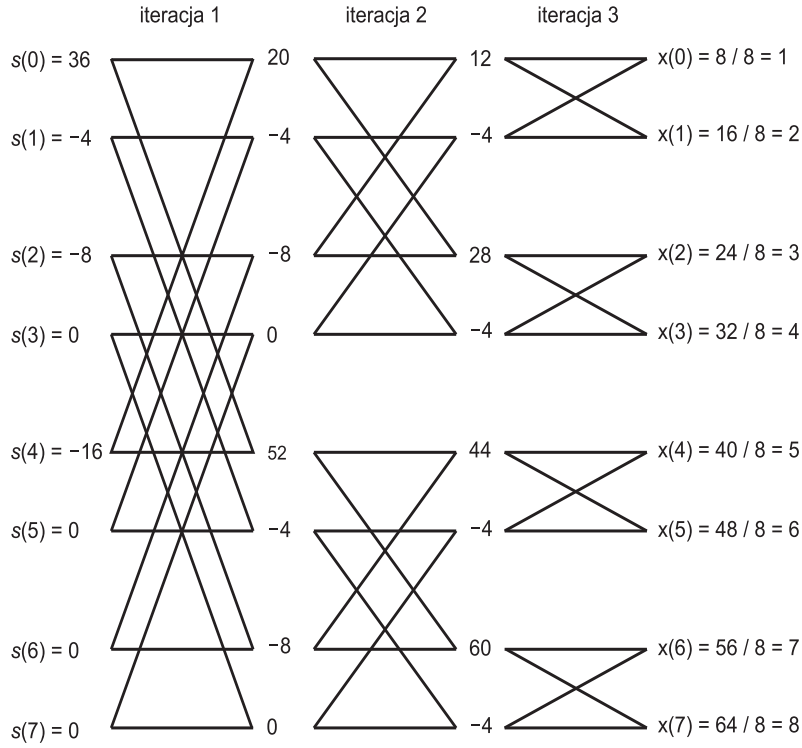
Inne rozwiązania wymagają jednak dodatkowych zabiegów związanych z odwracaniem bitów, co nie jest wygodne. Podobnie jak dla transformacji FFT, schematami motylkowymi można się posiłkować w programowaniu szybkich prostych i odwrotnych transformat z funkcjami Walsha. Ze względu na binarny charakter funkcji Washa oraz specyficzną budowę macierzy Hadamarda programowanie transformacji jest proste. Prostotę rozwiązania ilustruje poniższy przykład programu w języku Matlab.



Rys. 10. Schemat algorytmu szybkiej transformacji Walsha–Hadamarda dla $N = 8$ wraz z przykładem obliczeń w poszczególnych krokach

Program 5.18. *Program do wyznaczania prostej i odwrotnej szybkiej, jednowymiarowej transformacji Walsha–Hadamarda. Wersja szybka wymaga jednak, aby liczba N elementów ciągu była potęgą dwójki, $N \geq 2$.*

```
% Szybka transformacja Walsha-Hadamarda (wg schematu motylkowego)
rodzaj = 1;                     % 1 lub 2: transformacja
                                %  prosta lub odwrotna
A=[1 2 3 4];                    % Dane wejsciowe
FWHT(A,1)                       % Wlasna funkcja FWHT z parametrami
%% CIALO FUNKCJI  %%
function X=FWHT(dane,rodzaj)
% Zabezpieczenie programowe.Liczba N musi byc potega 2!!
N=pow2(floor(log2(length(dane))));
X = dane(1:N);                  % wektor danych
p1=2; p2=N/2; p3=1;             % Zmienne pomocnicze
for zm1=1:log2(N)               % Petla iteracji
    d1=1;
    for zm2 = 1:p2
        for zm3 = 1:p3
            i = zm3+d1-1;  j=i+p3;
           A= X(i);  B = X(j);
           X(i) = A + B;         % Elementarne operacje motylkowe
           X(j) = A - B;         % Elementarne operacje motylkowe
       end
       d1=d1+p1;
    end
    p1 = p1*2; p2 = p2/2; p3 = p3*2;
end
if (rodzaj==2)                % Rodzaj transformacji
   x=1/N*x;
end
```



Rys. 11. Schemat algorytmu szybkiej, odwrotnej transformacji Walsha–Hadamarda dla $N = 8$ wraz z przykładem obliczeń w poszczególnych krokach

5.8.4. Dyskretna dwuwymiarowa transformacja Walsha

Podobnie jak dla transformacji jednowymiarowych, można przedstawić zasady konstrukcji dwuwymiarowej transformacji Walsha. Prostym przykładem sygnału dwuwymiarowego może być na przykład obraz w formacie bitmapy.

Definicja 5.26. *Dwuwymiarową dyskretną transformacją Walsha* (*ang. 2D Discrete Walsh Transform – 2D-DWT*) *dla ciągu danych wejściowych* $x(m,n)$, $m = 0,1,...,M-1, \quad n = 0,1,...,N-1$ *nazywamy dwuwymiarowy ciąg współczynników rozwinięcia*:

$$s(k,l) = \alpha \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) \cdot wal(m,k) \cdot wal(n,l)$$
$$k = 0,1,...,M-1, \quad l = 0,1,...,N-1. \tag{5.173}$$

Podobnie opisać można odwrotną, dwuwymiarową transformację Walsha.

Definicja 5.27. *Dwuwymiarową odwrotną transformacją Walsha (ang. 2D Inverse Discrete Walsh Transform – 2D-IDWT) jest odwzorowanie, które na podstawie widma sygnału $s(k,l)$, $k = 0,...,M-1$, $l = 0,1,...,N-1$ odtwarza sygnał pierwotny $x(m,n)$:*

$$x(m,n) = \beta \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} s(k,l) \cdot wal(m,k) \cdot wal(n,l),$$
$$m = 0,1,...,M-1, \quad n = 0,1,...,N-1. \tag{5.174}$$

Operacje (5.173) oraz (5.174) są wzajemnie odwrotne, jeśli współczynniki skalowania związane będą zależnością:

$$\alpha \cdot \beta = \frac{1}{M \cdot N}. \tag{5.175}$$

Można wtedy stosować algorytmy tzw. obliczeń szybkich. Dwuwymiarowa transformacja Walsha realizowana jest przeważnie dla warunku $M = N$. Dla takiego ograniczenia otrzymujemy w zapisie macierzowym dwuwymiarową transformację Walsha–Hadamarda:

$$\boldsymbol{S} = \alpha \cdot \boldsymbol{H}_m \cdot \boldsymbol{X} \cdot \boldsymbol{H}_m, \tag{5.176}$$

$$\boldsymbol{X} = \beta \cdot \boldsymbol{H}_m \cdot \boldsymbol{S} \cdot \boldsymbol{H}_m, \tag{5.177}$$

gdzie $\boldsymbol{H}_m$, $m = \log_2 N$ jest odpowiednią macierzą Hadamarda o rozmiarze $2^m \times 2^m$.

Wyznaczenie wartości współczynników widmowych reprezentowanych przez elementy macierzy $\boldsymbol{S}$ zrealizować można bezpośrednio na podstawie wzorów (5.176), (5.177) lub za pomocą omówionych już wcześniej w tym rozdziale szybkich jednowymiarowych transformacji Walsha–Hadamarda.

Program 5.19. *Prosta i odwrotna dwuwymiarowa transformacja Walsha–Hadamarda. Rozwiązania programowe.*

```
A=round(2*rand(4,4)); % A jest macierza o wymiarach M X M
[M,N]=size(A);
% Zabezpieczenie programowe.Liczba M musi byc potega 2!!
% Zabezpieczenie programowe.Liczba N musi byc potega 2!!
M=pow2(floor(log2(M)));
N=pow2(floor(log2(N)));
% Wyznaczenie widma za pomoca zlozenia dwoch
% szybkich transformacji Walsha-Hadamarda 1D
widmo_1=FWHT(FWHT(A.',1).',1);
% Odtworzenie sygnalu pierwotnego
oryginal_1=FWHT(FWHT(widmo_1.',2).',2);
% Wykonanie zadania za pomoca funkcji bibliotecznej
% Odtworzenie sygnalu pierwotnego
widmo_2=hadamard(M)*A*(hadamard(N);
oryginal_2=1/(M*N)*hadamard(M)*widmo_2*hadamard(N);
```

Dwuwymiarową transformację Walsha–Hadamarda można również zorganizować według zasad zaprezentowanych w Programie 5.3. W tym celu występującą tam funkcję **Fast_F**() należy zastąpić funkcją **FWHT**() (z Programu 5.18), nie zmieniając parametrów, z którymi funkcje są wywoływane.

Oprócz klasycznie definiowanych funkcji Walsha, które jak pokazano są funkcjami odcinkowo-stałymi, stosowane są również funkcje, które uzyskuje się przez całkowanie funkcji Walsha. Otrzymujemy wtedy zbiór funkcji odcinkowo-liniowych [7, 9]. Funkcje odcinkowo-liniowe, nazywane także funkcjami WPL (ang. *Walsh Picewise Linear*), są bardzo przydatne w zagadnieniach aproksymacji sygnałów oraz w kompresji obrazów. Funkcje WPL charakteryzują się mniejszym błędem aproksymacji sygnału w porównaniu z klasycznymi funkcjami Walsha. Za pomocą funkcji WPL można również z dobrym skutkiem kompresować obrazy [9], gdzie klasyczne funkcje Walsha zawodzą.

Dla funkcji WPL istnieją proste i odwrotne jedno- oraz dwuwymiarowe transformacje, których algorytmy wraz z odpowiednim programem w Matlabie są opisane w pracy [9]. Z tych powodów nie są tutaj przytaczane. Zainteresowany teorią Czytelnik znajdzie szczegółowy opis tych zagadnień w cytowanej powyżej pracy.

5.8.5. Binarne funkcje Walsha

Dyskretne funkcje Walsha mają również inną reprezentację niż omawiana poprzednio. Zamiast elementów $\{+1, -1\}$ mogą być one opisywane za pomocą elementów $\{0, 1\}$, zgodnie z odwzorowaniem $\{+1, -1\} \rightarrow \{0, 1\}$. Tym samym zamiast klasycznej macierzy Hadamarda $\boldsymbol{H}_m$ otrzymujemy macierz binarną $\boldsymbol{H}_m^b$: $\boldsymbol{H}_m \overset{binaryzacja}{\rightarrow} \boldsymbol{H}_m^b$. Na przykład dla $m = 2$ otrzymujemy:

$$\boldsymbol{H}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \overset{binaryzacja}{\longrightarrow} \boldsymbol{H}_2^b = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}. \tag{5.178}$$

Podobnie jak macierz $\boldsymbol{H}_m$, macierz $\boldsymbol{H}_m^b$ jest macierzą symetryczną. Ponieważ wyznacznik binarnej macierzy Walsha $det(\boldsymbol{H}_m^b) = 0$, macierz tego typu nie jest macierzą odwracalną. Wiersze lub kolumny binarnej macierzy $\boldsymbol{H}_m^b$ można utożsamiać z wektorem prawdy m-argumentowej w pełni określonej funkcji boolowskiej. Są to tzw. liniowe funkcje boolowskie. Każda macierz typu $\boldsymbol{H}_m^b$ zawiera zawsze 2^m wektorów prawdy opisujących boolowskie funkcje liniowe. Binarne funkcje Washa oznaczane będą $wal^b(j, x)$. Można zauważyć, że dowolną funkcję Walsha $wal^b(j, x)$ można przedstawić jako sumę modulo 2 innych binarnych funkcji Walsha:

$$wal^b(j, x) = \bigoplus_{k=0}^{m-1} b_k wal^b(2^k, x), \tag{5.179}$$

gdzie b_k jest bieżącą wartością pozycji binarnego rozwinięcia liczby j, a m jest liczbą pozycji tego rozwinięcia.

Przykład 5.15. *Dla liczby* $j = 23$ *i jej binarnej reprezentacji* $(23)_{10} = (10111)_2$, *otrzymujemy*:

$$\begin{aligned} wal^b(23,x) &= \bigoplus_{k=0}^{4} b_k wal(2^k,x) = \\ &= wal^b(1,x) \oplus wal^b(2,x) \oplus wal^b(4,x) \oplus wal^b(16,x). \end{aligned} \tag{5.180}$$

Dla dyskretnych funkcji Walsha z elementami 0,1 zachodzi również zależność:

$$wal^b(i,x) \oplus wal^b(j,x) = wal^b(i \oplus j, x), \tag{5.181}$$

gdzie symbol $\oplus$ jest sumą modulo 2 wyrażonych binarnie numerów funkcji.

Tak więc wzory (5.155) i (5.181) są równoważne w takim sensie, że mnożenie funkcji Walsha z elementami $+1$ oraz -1 jest równoważne ich sumowaniu modulo 2 przy elementach 0 i 1. Można również zaobserwować, że zapis (5.179) jest tylko jednym ze sposobów tworzenia funkcji Walsha, gdyż na podstawie (5.181) istnieje duża swoboda w doborze funkcji składowych tworzących wypadkową funkcję Walsha:

$$wal^b(23,x) = wal^b(6,x) \oplus wal^b(17,x), \tag{5.182}$$

ale również:

$$wal^b(23,x) = wal^b(3,x) \oplus wal^b(20,x) = wal^b(24,x) \oplus wal^b(15,x). \tag{5.183}$$

Każda binarna funkcja Walsha $wal^b(j,x)$ ma swój odpowiednik $wal(j,x)$, co wydaje się oczywiste w świetle przytoczonych wywodów. W tym celu należy dokonać prostego przekodowania wszystkich wartości binarnych funkcji $wal^b(j,x)$, zgodnie z formułą: $\{0,1\} \to \{1,-1\}$.

5.9. Funkcje i transformacje Haara

System ortogonalnych funkcji Haara został przedstawiony przez A. Haara w 1910 r. i dotyczył funkcji definiowanych w przedziale $[0,1)$. Funkcje Haara są często opisywane w bardzo różny sposób. Oto jedna z definicji:

Definicja 5.28. *Ortogonormalny zbiór funkcji Haara* $Har(m,n,t)$ *jest zbudowany następująco* [28]:

$$Har(0,0,t) = 1,\ t \in [0,1), \tag{5.184}$$

$$Har(r,m,t) = \begin{cases} 2^{r/2} & \dfrac{m-1}{2^r} \le t < \dfrac{m-\frac{1}{2}}{2^r} \\ -2^{r/2} & \dfrac{m-\frac{1}{2}}{2^r} \le t < \dfrac{m}{2^r} \\ 0 & \textit{dla pozostałych } t \in [0,1) \end{cases}, \tag{5.185}$$

gdzie: $0 \le r < \log_2 N$ *oraz* $1 \le m \le 2^r$.

Wykres pierwszych ośmiu funkcji Haara zaprezentowano na rys. 12.
W praktyce sposób konstrukcji funkcji Haara oparty jest na sukcesywnym podziale przedziałów określoności funkcji, w których funkcja ma zwarty nośnik.

Definicja 5.29. *Nośnikiem funkcji ciągłej* $f : \boldsymbol{R}^m \to \boldsymbol{R}$ *nazywamy zbiór takich* $x \in \boldsymbol{R}^m$, *dla których* $f(x) \neq 0$. *Nośnik funkcji oznaczany jest symbolem supp* f, *zatem*:

$$supp\ f = \{x \in \boldsymbol{R}^m : f(x) \neq 0\}. \tag{5.186}$$

Zgodnie z Definicją 5.29, nośnik funkcji zawiera te przedziały, w których funkcje elementarne są niezerowe. Jeżeli te przedziały są również domknięte, to nośnik jest nośnikiem zwartym. Oznacza to, że czas trwania funkcji f jest skończony.

Przedział $[0,1)$, na którym określona jest funkcja $Har(0,0,t)$, dzielony jest na dwa podprzedziały, z których konstruowana jest funkcja $Har(0,1,t)$. Następnie każdy z dwóch podprzedziałów jest traktowany oddzielnie i dzielony na kolejne dwa podprzedziały, w których konstruowane są następne funkcje Haara. Procedura podziału jest kontynuowana do momentu otrzymania wszystkich 2^m funkcji Haara. Ortonormalność tych funcji można wykazać podobnie jak dla funkcji Walsha. Funkcje Haara $Har(r,m,t)$ można także opisywać za pomocą jednego parametru. Zamiast dotychczasowych dwóch parametrów r oraz m można zastosować jeden parametr $i = 2^r + m - 1$.

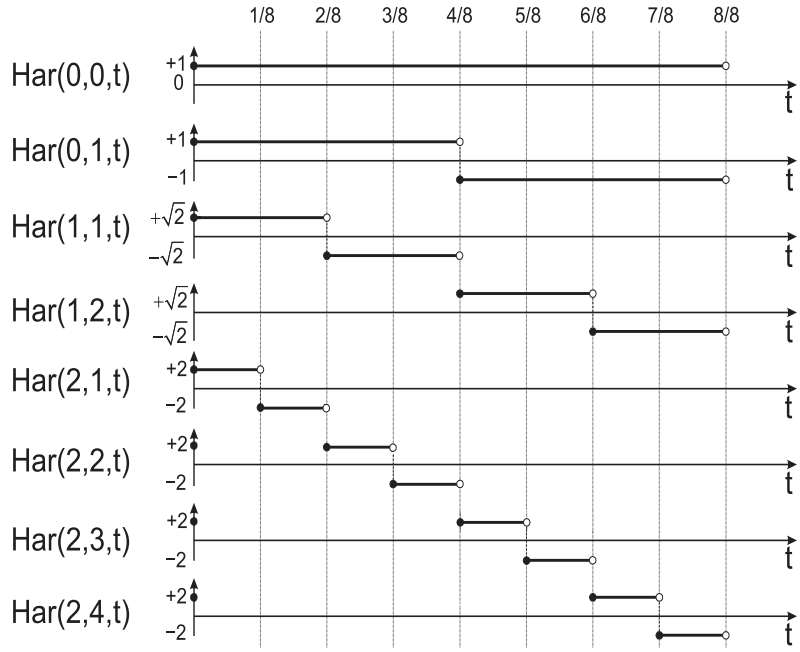
W takim przypadku regułę wiążącą oba zapisy funkcji Haara przedstawić można następująco:

$$Har(r, m, t) = Har(i, t). \tag{5.187}$$

Oba zapisy są równoważne i dotyczą tych samych funkcji Haara. Korzystając z jednoparametrowego opisu funkcji Haara, można je konstruować, stosując operacje przesuwania i zmiany skali argumentu funkcji podstawowej $Har(1, t)$:

$$Har(r, m, t) = \sqrt{2^r} Har(1, 2^r t - m - 1), \tag{5.188}$$

gdzie $t \in [0, 1)$.



Rys. 12. Wykres pierwszych ośmiu ($N = 8$) funkcji Haara określonych na przedziale $[0, 1)$. UWAGA! Oś rzędnych nie jest skalowana

5.9.1. Dyskretne funkcje Haara

Dyskretne funkcje Haara otrzymujemy w rezultacie dyskretyzacji ciągłych funkcji Haara, zawierających skończoną liczbę punktów nieciągłości w przedziale $[0,1)$. Dyskretyzacja przeprowadzana jest w ten sposób, że cały przedział określoności każdej funkcji Haara dzielony jest na N elementarnych podprzedziałów i w środku każdego podprzedziału $\left[\frac{n}{N}, \frac{n+1}{N}\right)$, $n = 0, ..., N-1$, $N = 2^m$ pobierana jest wartość funkcji Haara.

Zbiór utworzonych w ten sposób dyskretnych punktów można przedstawić w postaci macierzy. W literaturze przywoływany jest często przykład macierzy 8×8 [9,28], gdzie można zauważyć przedziałowe podobieństwa do funkcji Haara z rys. 12.

$$\boldsymbol{H}r_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}. \tag{5.189}$$

Każdy wiersz macierzy $\boldsymbol{H}r_m$ może być utożsamiany z dyskretną funkcją Haara. Tym samym funkcje te można również rozpatrywać jako wektory wierszowe o N współrzędnych. Stosując oznaczenia jak we wzorach (5.185) oraz (5.189), dyskretne funkcje Haara można wyznaczać rekurencyjnie z blokowo zdefiniowanej macierzy $\boldsymbol{H}r_m$:

$$\boldsymbol{H}r_m = \begin{bmatrix} \boldsymbol{H}r_{m-1} & \otimes \begin{bmatrix} +1 & +1 \end{bmatrix} \\ 2^{(m-1)/2}\boldsymbol{I}_{2^{m-1}} & \otimes \begin{bmatrix} +1 & -1 \end{bmatrix} \end{bmatrix}, \quad \boldsymbol{H}r_0 = [1], \quad m = 1, 2, ... \tag{5.190}$$

gdzie:

$\boldsymbol{H}r_m$ – macierz $2^m \times 2^m$,

$\boldsymbol{I}_m$ – macierz jednostkowa $2^m \times 2^m$,

$\otimes$ – iloczyn Kroneckera.

Wiersze macierzy $\boldsymbol{H}r_m$, $m = 1, 2, ... \log_2(N)$ są parami ortogonalne i każdy z nich można unormować do jedności:

$$\widetilde{\boldsymbol{H}}r_m = \frac{1}{\sqrt{N}}\boldsymbol{H}r_m. \tag{5.191}$$

$$\boldsymbol{H}r_m \neq (\boldsymbol{H}r_m)^T \quad \text{oraz} \quad (\boldsymbol{H}r_m)^{-1} = \frac{1}{N}(\boldsymbol{H}r_m)^T. \tag{5.192}$$

Dyskretne funkcje Haara opisane macierzą $\boldsymbol{H}r_m$ są ortogonalne i nieunormowane, co można wykazać analogicznie jak w przypadku funkcji Walsha. Macierze $\boldsymbol{H}r_m$ większych rozmiarów można wygenerować programowo.

Program 5.20. *Funkcja w programie Matlab do generowania macierzy Haara* $\boldsymbol{H}r_m$ *o rozmiarach* $2^m \times 2^m$.

```
Function [Hr]=Haar_M(N);     % Liczba N musi byc potega liczby 2
% Zabezpieczenie programowe. Liczba N musi byc potega 2!!
N=pow2(floor(log2(N))); Hr=[1];
p=ceil(log2(N));             % Wyznaczenie liczby iteracji
for m=1:p                    % Macierz Hr o wymiarach N x N
    A=kron(Hr,[1 1]);       % Pierwszy wiersz macierzy blokowej Hr
    B = kron(2^((m-1)/2)*eye(2^(m-1)),[1 -1]);
    Hr = [A;B];              % Kompletna macierz Haara
end
```

Z zależności (5.192) wynika, że macierz odwrotnej transformacji Haara różni się od macierzy transformacji prostej – macierz $\boldsymbol{H}r_m$ nie jest bowiem macierzą symetryczną. Z tego powodu algorytmy prostej i odwrotnej transformacji Haara różnią się od siebie. Stanowi to niedogodność, szczególnie przy zapisywaniu tych algorytmów w językach programowania. Utrudnienia tego typu nie występują w algorytmach prostej i odwrotnej transformacji Walsha, gdzie można

wykorzystywać ten sam schemat obliczeń zarówno dla prostej, jak i odwrotnej transformacji. Niedogodności te powodują, że spotkać można wiele przykładów algorytmów realizujących transformacje szybkie według różnych założeń.

5.9.2. Dyskretna jednowymiarowa transformacja Haara

Macierz Haara $\boldsymbol{H}r_m$ nie jest macierzą symetryczną, tak jak to miało miejsce np. w przypadku macierzy Walsha, dlatego sposób przeprowadzania transformacji ma wpływ na uzyskiwane wyniki obliczeń.

Definicja 5.30. *Jednowymiarową dyskretną transformacją Haara* (*ang. Discrete Haar Transform – DHT*) *sygnału* $x(n)$ *określonego w chwilach* $n = 0, 1, ..., N-1$, $N = 2^m$ *nazywamy odwzorowanie, w wyniku którego otrzymujemy następującą transformatę*:

$$s(k) = \alpha \sum_{k=0}^{N-1} haar(n,k) \cdot x(n), \qquad k = 0, 1, ..., N-1, \tag{5.193}$$

gdzie $haar(n,k)$ *jest n-tą dyskretną funkcją Haara zapisaną w n-tym wierszu macierzy* $\boldsymbol{H}r_m$.

Definicja 5.31. *Jednowymiarowa dyskretna odwrotna transformacja Haara* (*ang. Inverse Discrete Haar Transform – IDHT*) *jest odwzorowaniem, które na podstawie widma sygnału* $s(k)$, $k = 0, ..., N-1$ *odtwarza sygnał pierwotny* $x(n)$, $n = 0, ..., N-1$:

$$x(n) = \beta \sum_{n=0}^{N-1} haar(n,k) \cdot s(k), \qquad n = 0, 1, ..., N-1, \tag{5.194}$$

gdzie $haar(n,k)$ *jest n-tą dyskretną funkcją Haara zapisaną w n-tym wierszu macierzy* $\boldsymbol{H}r_m^T$.

Porównując definicje jednowymiarowych transformacji Walsha i Haara, można zauważyć, że te ostatnie zostały przedstawione nieco inaczej. Dla transformacji Walsha dokonywano mnożenia wektora danych przez funkcje bazowe. Teraz postąpiono odwrotnie – mnoży się funkcje bazowe przez wektor danych.

Jest to zabieg celowy, chociaż forma zapisu wymaga, aby zasady definiowania transformacji były jednakowe. Odstępstwo to spowodowane jest wyłącznie celem dydaktycznym – nie ma to znaczenia praktycznego – każda z definicji jest poprawna.
Transformacje (5.193) oraz (5.194) przedstawić można w postaci macierzowej:

$$\boldsymbol{s} = \alpha \cdot \boldsymbol{H}r_m \cdot \boldsymbol{x}. \tag{5.195}$$

Zgodnie z zależnościami (5.192), można zapisać:

$$\boldsymbol{x} = \beta \cdot (\boldsymbol{H}r_m)^T \cdot \boldsymbol{s}. \tag{5.196}$$

Wyznaczyć prostą, a następnie odwrotną transformację Haara wektora danych $\boldsymbol{x} = [1, 2, -2, 0]^T$, korzystając w tym celu ze wzorów (5.195) oraz (5.196). $N = 4$, $m = log_2 N = 2$. Przyjmując $\alpha = 1$ oraz $\beta = \frac{1}{4}$, otrzymujemy:

$$\boldsymbol{s} = \boldsymbol{H}r_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ -2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ -\sqrt{2} \\ -2\sqrt{2} \end{bmatrix}, \tag{5.197}$$

$$\boldsymbol{x} = \frac{1}{4} \cdot (\boldsymbol{H}r_2)^T \cdot \boldsymbol{s} = \frac{1}{4} \cdot \begin{bmatrix} 1 & 1 & \sqrt{2} & 0 \\ 1 & 1 & -\sqrt{2} & 0 \\ 1 & -1 & 0 & \sqrt{2} \\ 1 & -1 & 0 & -\sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 5 \\ -\sqrt{2} \\ -2\sqrt{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -2 \\ 0 \end{bmatrix}. \tag{5.198}$$

5.9.3. Dyskretne szybkie transformacje Haara

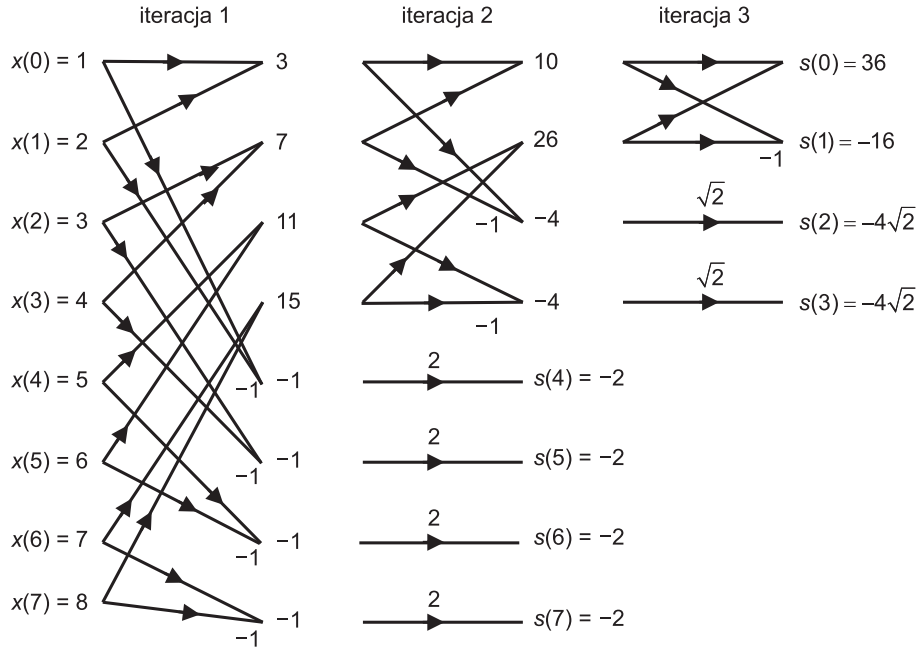
Algorytm szybkiej transformacji Haara korzysta z możliwości zastąpienia klasycznego mnożenia macierzy metodą, w której wszystkie operacje arytmetyczne zredukowane zostają do sumowania i dodawania. Idea transformacji szybkich może być implementowana na wiele sposobów. Tutaj zostaną przedstawione dwa najczęściej spotykane rozwiązania.

Pierwszy z algorytmów prostej (Program 5.19) oraz odwrotnej (Program 5.20) transformacji Haara został opracowany przez Andrewsa [9,28]. Drugi ma budowę podobną do algorytmu szybkiej transformacji Fouriera. Jest to algorytm wzorowany na metodzie Cooleya–Tukeya dla algorytmu FFT [11,28]. Oba wymienione typy algorytmów dają oczywiście te same wyniki. Przedstawiona tutaj wersja algorytmu Andrewsa wymaga wykonania $2(N-1)$ operacji dodawania i odejmowania oraz $N-1$ operacji mnożenia, co można łatwo sprawdzić, posługując się rys. 13.

Oznacza to, że złożoność tego algorytmu jest liniowa i wynosi tylko $O(N)$, co jest wynikiem lepszym niż w algorytmach typu FFT o złożoności $O(N \log_2 N)$. Można również sprawdzić, że w algorytmie prostej i odwrotnej transformacji Haara sumaryczna liczba niezbędnych do wykonania operacji dodawania, odejmowania oraz mnożenia jest taka sama i wynosi $2(N-1)$.

Schematy Andrewsa prostej i odwrotnej transformacji Haara są więc różne, co utrudnia programową implementację tych algorytmów. Przedstawione niedogodności skłoniły do poszukiwań rozwiązań, w których można stosować schematy motylkowe – znane z algorytmu FFT lub szybkiej transformacji Walsha. Algorytmy oparte na schematach motylkowych zachowują tę samą zasadę obliczeń dla prostej i odwrotnej transformacji, co jest dużym udogodnieniem. Zasadę działania wymienionych algorytmów można pokazać na przykładach odpowiednich schematów przepływowych.

Poniżej przedstawiono szczegółowo zarówno graficzny opis odpowiednich algorytmów, jak i ich implementacje programowe w języku Matlab. W celach poglądowych na wszystkich rysunkach zamieszczono również cząstkowe wyniki obliczeń uzyskiwane w kolejnych krokach iteracji. Obliczenia widma Haara dotyczą arbitralnie wybranego ośmioelementowego, wejściowego wektora danych.



Rys. 13. Schemat algorytmu szybkiej transformacji Haara (wersja Andrewsa) dla $N = 8$ wraz z przykładem obliczeń w poszczególnych krokach

Program 5.21. *Program szybkiej transformacji Haara zbudowany na podstawie grafu przepływowego z rys.* 13.

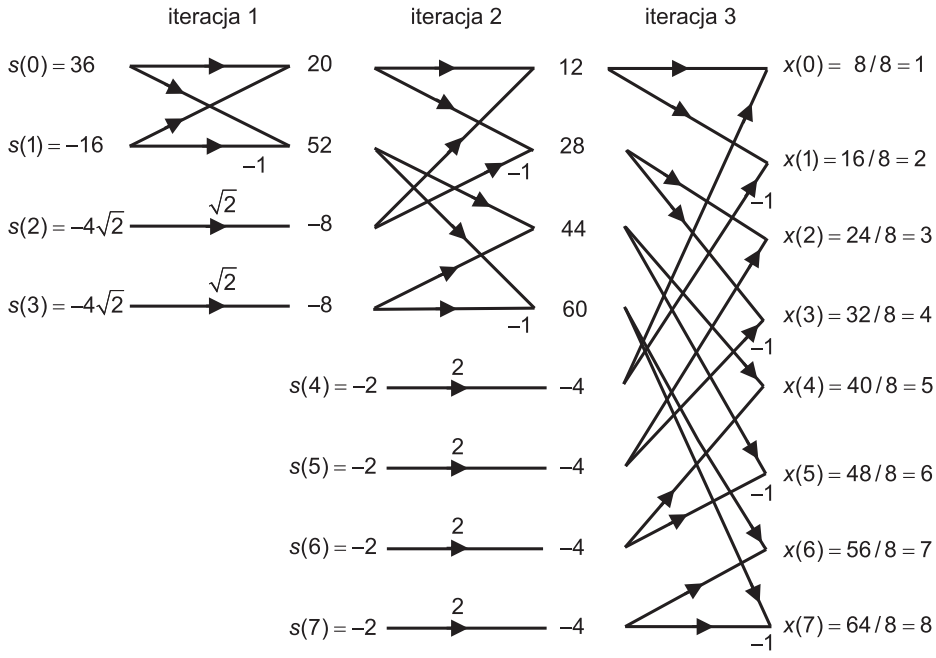
```
A=[1 2 3 4];              % Przykladowe dane wejsciowe
Haar_A(A);                % Wywolanie funkcji Haar_A(wersja Andrewsa)
%% CIALO FUNKCJI  %%
function X=Haar_A(dane)
N=pow2(floor(log2(length(dane)))); % Liczba N musi byc potega 2!!
X=dane(1:N);              % wektor danych
m=N;                      % Zmienna pomocnicza
while(1<m)                % Petla iteracji
      m=floor(m/2);
      w(1:m)=X(1:2:2*m-1)+X(2:2:2*m);
      w(m+1:m+m)=X(1:2:2*m-1)-X(2:2:2*m);
```

```
      X(1:2*m)=w(1:2*m);
end
for i2=1:log2(N)-1
    p=2^i2;
    A=X(p+1:2^(i2+1));
    c=length(A);
    P=(2^(i2/2));     W=A*P;
    for i3=1:c
        X(p+i3)=W(i3);
    end
end
```

Program 5.22. *Program szybkiej, odwrotnej transformacji Haara zbudowany na podstawie grafu przepływowego z rys.* 14.

```
A=[1 2 3 4];        % Przykladowe dane wejsciowe (widmo)
IHaar_A(A);         % Wywolanie funkcji IHaar_A (wersja Andrewsa)
%%  CIALO FUNKCJI  %%
function x=IHaar_A(dane)
N=pow2(floor(log2(length(dane)))); % Liczba N musi byc potega 2!!
X=dane(1:N);        % Probki wektora danych
m=1;                % Zmienna pomocnicza
for i=1:log2(N)     % Petla iteracji
    w(1:2:2*m-1)=X(1:m)+X(1+m:2*m);
    w(2:2:2*m)=X(1:m)-X(1+m:2*m);
    X(1:2*m)=w(1:2*m);
    if i<log2(N)
       s=2^(i/2);
       X(2*m+1:2*m+2^i)=s*X(2*m+1:2*m+2^i);
    end
    m=m*2;
end
X=X/N;
```

Jak pokazano, transformacje Haara można realizować na podstawie schematu motylkowego, co pozwala ujednolicić obliczenia. Podobnie jak w klasycznej transformacji FFT, zachodzi jednak konieczność przemieszczania próbek na inne pozycje. Należy zauważyć, że w transformacji Walsha wykorzystującej schemat motylkowy odwracanie bitów nie było potrzebne, co jest bardzo dużym udogodnieniem upraszczającym programowanie algorytmów. Jest to kolejny przykład celowości poszukiwań prostych metod transformacji danych. Schematy motylkowe algorytmów prostej i odwrotnej szybkiej transformacji Haara przedstawiono na rys. 15 i rys. 16. Dla tych schematów zaprezentowano także odpowiednie programy komputerowe.

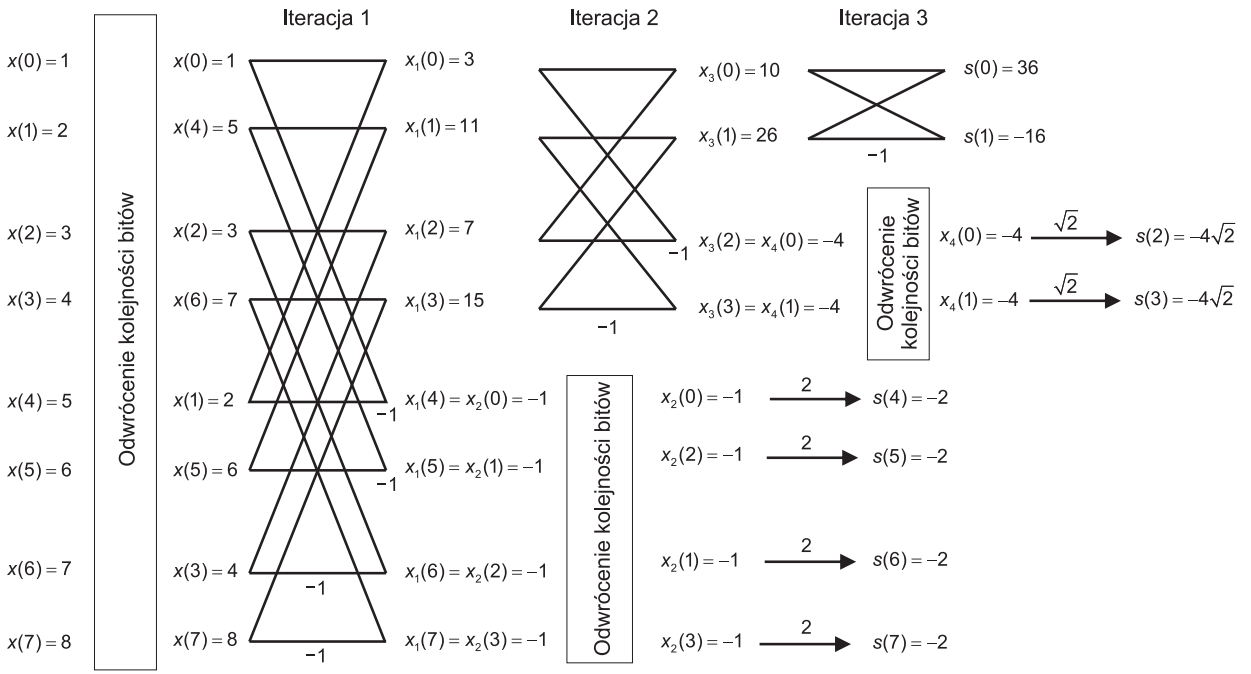


Rys. 14. Schemat algorytmu szybkiej, odwrotnej transformacji Haara (wersja Andrewsa) dla $N = 8$ wraz z przykładem obliczeń w poszczególnych krokach

Program 5.23. *Program szybkiej, jednowymiarowej transformacji Haara zrealizowny na podstawie schematu motylkowego przedstawionego na rys.* 15.

```
% Szybka transformacja Haara
% Wersja programowana wg schematu motylkowego
A=[1 2 3 4];                % Przykladowe dane wejsciowe
Haar_F(A)                   % Wywolanie wlasnej funkcji Haar_F
%%  CIALO FUNKCJI   %%
% Wywolanie funkcji Haar_F (wersja motylkowa)
function X=Haar_F(dane)
N=pow2(floor(log2(length(dane)))); % Liczba N musi byc potega 2!!
% Binarna numeracja elementow wektora wejsciowego
I=dec2bin(0:pow2(0.5,e)-1);
% Odwrocenie kolejnosci bitow. Nowy porzadek danych
R=dane(bin2dec(I(:,e-1:-1:1))+1);
dane=R; .
X=dane(1:N);              % Uporzadkowane probki wektora danych
k1=N;
k2=N/2;
for ip=1:log2(N)
    for i1=1:k2;
        i=i1;
        j=i+k2;
        temp1=X(i);
        temp2=X(j);
        X(i)=temp1+temp2;    % Elementarne operacje motylkowe
        X(j)=temp1-temp2;    % Elementarne operacje motylkowe
    end
    k1=k1/2;
    k2 = k2/2;
end
for i2=1:log2(N-1);
```

```
    p=2^i2;
    A=X(p+1:2^(i2+1));
    c=length(A);
    [f,e]=log2(length(A));
    I=dec2bin(0:pow2(0.5,e)-1);
    R=A(bin2dec(I(:,e-1:-1:1))+1);
    P=(2^(i2/2));
    W=R*P;
    for i3=1:c
        X(p+i3)=W(i3);
    end
end
```



Rys. 15. Schemat motylkowy algorytmu szybkiej transformacji Haara dla $N = 8$ wraz z wynikami obliczeń w poszczególnych krokach

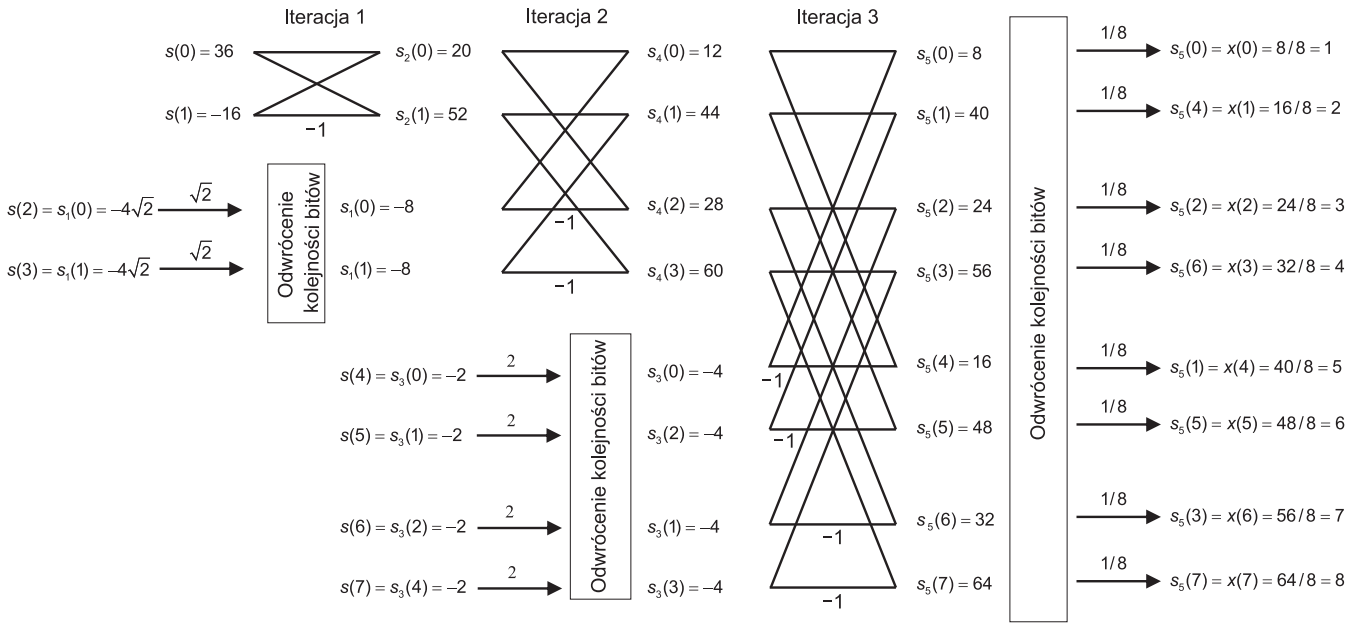
Podobny schemat motylkowy (rys. 16) można zastosować w algorytmie szybkiej, odwrotnej transformacji Haara.

Program 5.24. *Program jednowymiarowej, szybkiej, odwrotnej transformacji Haara zbudowany na podstawie motylkowego grafu przepływowego z rys.* 16.

```
% Szybka, odwrotna transformacja Haara
% Wersja programowana wg schematu motylkowego
A=[1 2 3 4];        % Przykladowe dane wejsciowe
IHaar_F(A)          % Wywolanie wlasnej funkcji IHaar_F
%%  CIALO FUNKCJI   %%
% Wywolanie funkcji IHaar_F (wersja motylkowa)
function X=IHaar_F(dane)
N=pow2(floor(log2(length(dane)))); % Liczba N musi byc potega 2!!
k1=1;
for ip=1:log2(N)
    for i1=1:k1;
        i=i1;
        j=i+k1;
        temp1=X(i);
        temp2=X(j);
        X(i)=temp1+temp2;   % Elementarne operacje motylkowe
        X(j)=temp1-temp2;   % Elementarne operacje motylkowe
    end
    if ip < log2(N)
        p=2^ip;
        A=X(j+1:2*j);
        c=length(A);
        [f,e]=log2(length(A));
        % Binarna numeracja elementow wektora wejsciowego
        I=dec2bin(0:pow2(0.5,e)-1);
        % Odwrocenie kolejnosci bitow. Nowy porzadek danych
```

```
        R=A(bin2dec(I(:,e-1:-1:1))+1);
        P=(2^(ip/2));
        W=R*P;
        for i2=1:c
            X(p+i2)=W(i2);
       end
       k1=k1*2;
    end
end
c=length(X);
[f,e]=log2(length(X));
I=dec2bin(0:pow2(0.5,e)-1);
R=X(bin2dec(I(:,e-1:-1:1))+1);   X=X/N;
```

Można zauważyć, że ze względu na własności funkcji Haara schematy przepływowe prostej i odwrotnej transformacji Haara są odmienne.



Rys. 16. Schemat motylkowy algorytmu szybkiej, odwrotnej transformacji Haara dla $N = 8$ wraz z przykładem obliczeń w poszczególnych krokach

5.9.4. Dyskretna dwuwymiarowa transformacja Haara

Dwuwymiarową prostą i odwrotną transformację Haara można zdefiniować następująco:

Definicja 5.32. *Dwuwymiarową dyskretną transformacją Haara (ang 2D. Discrete Haar Transform - 2D-DHT) ciągu* $x(m,n)$,$m = 0,1,...,M-1, \quad n = 0,1,...,N-1$ *nazywamy odwzorowanie, w wyniku którego otrzymujemy następującą transformatę*:

$$\begin{aligned} s(k,l) &= \alpha \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} haar(m,k) \cdot haar(n,l) \cdot x(m,n), \\ k, &= 0,1,...,M-1, \quad l = 0,1,...,N-1. \end{aligned} \tag{5.199}$$

Definicja 5.33. *Dwuwymiarową odwrotną transformacją Haara (ang. 2D Inverse Discrete Haar Transform - 2D-IDHT) jest odwzorowanie, które na podstawie widma sygnału* $s(k,l)$, $k,0,...,M-1$, $l = 0,1,...,N-1$ *odtwarza sygnał pierwotny* $x(m,n)$:

$$\begin{aligned} x(m,n) &= \beta \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} haar(m,k) \cdot haar(n,l) \cdot s(k,l), \\ m &= 0,1,...,M-1, \quad n = 0,1,...,N-1. \end{aligned} \tag{5.200}$$

Ze względów praktycznych można wtedy stosować algorytmy tzw. obliczeń szybkich – dwuwymiarowa transformacja Haara realizowana jest przeważnie dla warunku $M = N$. Dla takiego ograniczenia otrzymujemy w zapisie macierzowym prostą i odwrotną dwuwymiarową transformację Haara:

$$\boldsymbol{S} = \alpha \cdot \boldsymbol{H}r_m \cdot \boldsymbol{X} \cdot \boldsymbol{H}r_m^T, \tag{5.201}$$

$$\boldsymbol{X} = \beta \cdot \boldsymbol{H}r_m^T \cdot \boldsymbol{S} \cdot \boldsymbol{H}r_m, \tag{5.202}$$

gdzie powiązanie między współczynnikami skalowania musi być następujące:

$$\alpha \cdot \beta = \frac{1}{M \cdot N}. \tag{5.203}$$

Program dwuwymiarowej transformacji Haara można zorganizować na podobnych zasadach jak zrobiono to w przypadku opisywanych już wcześniej transformacji Fouriera i Walsha.

Program 5.25. *Prosta i odwrotna dwuwymiarowa transformacja Haara. W programie wykorzystano własną funkcję biblioteczną* **Haar_M**(), *stosowaną już poprzednio w Programie* (5.20).

```
A=round(2*rand(4,4));          % A jest macierza o wymiarach M X M
[M,N]=size(A);                 % Wyznaczenie rozmiarow
                               % M oraz N macierzy A
M=pow2(floor(log2(M)));        % Liczba M musi byc potega 2
N=pow2(floor(log2(N)));        % Liczba N musi byc potega 2
% Wykonanie zadania za pomoca funkcji bibliotecznej
widmo_1=Haar_M(M)*A*Haar_M(N)';
% Odtworzenie sygnalu pierwotnego
oryginal_2=1/(M*N)*Haar_M(M)'*widmo_1*Haar_M(N);
```

6. Wybrane zastosowania dyskretnego przetwarzania danych

Z przekształcenia sygnału pierwotnego otrzymujemy jego nową reprezentację – transformatę sygnału. Podstawowym celem przekształcenia jest podkreślenie istotnych cech charakterystycznych sygnału pierwotnego. Szybki rozwój algorytmów cyfrowego przetwarzania sygnałów powoduje, że techniki transformacji są wykorzystywane prawie we wszystkich dziedzinach techniki. Wydaje się, że w ostatnich latach najwięcej metod cyfrowego przetwarzania danych znalazło zastosowanie w informatyce, medycynie, biologii i fizyce, gdzie były wykorzystywane do usuwania zakłóceń i zniekształceń na obrazach medycznych lub biologicznych, a także do wydobywania i obrazowania elementów lub parametrów obrazu, które będą najbardziej przydatne z punktu widzenia lekarza diagnosty lub naukowca badacza. Transformacje sygnałów dyskretnych znajdują również zastosowanie w elektronice – dla optymalizacji i diagnostyki układów cyfrowych [24,40]. Obserwacja danych w transformowanych przestrzeniach umożliwia na przykład wykrycie periodyczności w obrazie albo związków występujących w funkcjach boolowskich, ułatwiając dekompozycję funkcji lub ich opis strukturalny w postaci Binarnych Diagramów Decyzyjnych [18,19,24].

Dysponując odpowiednimi danymi, możemy poddawać je różnym transformacjom, które omawiane były w poprzednich rozdziałach. Dla różnych danych preferowane są różne transformacje, a więc ich znajomość wydaje się ważna. Za pomocą odpowiednich operacji na obrazie cyfrowym można tłumić, wzmacniać lub filtrować elementy obrazu, analizować wybrane składowe, odszumiać lub pozbywać się atefaktów [35,40]. Można również odkrywać specyficzny rozkład danych, co pozwala je klasyfikować, lub nawet wykrywać zawarte w tych danych błędy [24]. Transformacje sygnałów poprzedzone są często skomplikowaną procedurą doprowadzającą dane do postaci cyfrowej. Sygnał analogowy, na przykład w postaci natężenia światła, temperatury lub ciśnienia akustycznego, jest przetwarzany za pomocą odpowiedniego przetwornika analogowo-cyfrowego.

Zadaniem przetwornika jest zamiana sygnału analogowego (ciągłego) na postać cyfrową. Proces ten polega na zastąpieniu płynnie zmieniających się wartości wartościami zmieniającymi się skokowo w odpowiedniej skali odwzorowania.

Przykładem zastosowania przetwarzania analogowo-cyfrowego jest skaner, w którym analogowy obraz przekształcany jest do postaci bitmapy, co oznacza, że powierzchnia obrazu zostaje podzielona na odpowiednią liczbę pikseli. Takie przetworniki wykorzystywane są również w kamerach cyfrowych, tomografii komputerowej, obrazowaniu za pomocą rezonansu magnetycznego, elektrokardiografii, ultrasonografii, okulistyce, mikroskopii elektronowej i wielu innych. Nie jest to jednak przedmiotem naszych rozważań.

Niektóre metody odkrywania specyficznych związków występujących w danych dyskretnych będą przedstawione poniżej. Należy jednak zwrócić uwagę Czytelnika na fakt, że odkrywanie takich związków lub cech oraz ich klasyfikacja są zagadnieniami szeroko opisywanymi w literaturze, zarówno z pozycji teorii, jak i algorytmów. Szeroki wachlarz tych metod pozwala, przynajmniej intuicyjnie, na zrozumienie mechanizmów analizy sygnałów. Mechnizmy takie w warstwie teoretycznej i algorytmicznej (kompletne programy w Matlabie) były już prezentowane w niniejszej książce. Aplikacje praktyczne są jedynie konsekwencją rozwiązań wypracowanych przez matematyków, automatyków, elektroników czy informatyków razem z ekspertami z danej dziedziny. W tym rozdziale przedstawione zostały autorskie rozwiązania powiązane z dyskretnym przetwarzaniem danych. Stanowią one encyklopedyczne i porządkujące streszczenie zagadnień teoretycznych opisywanych już w rozdziałach poprzednich.

6.1. Transformacje w zastosowaniach przetwarzania obrazów 2D

Podobnie jak w przypadku sygnałów jednowymiarowych, sygnały wielowymiarowe mogą występować w postaci ciągłej lub dyskretnej. Analizując opisywane już dwuwymiarowe transformacje Fouriera w bazach sinusów (DST), kosinusów (DCT), Walsha i Haara, można sprawdzić, jak kształtują się widma sygnału dwuwymiarowego w tych bazach.

Obraz wyświetlany na monitorze komputera może być traktowany jako funkcja dwóch zmiennych – dyskretnych współrzędnych obrazu. Obserwacja widma sygnału dwuwymiarowego może być kryterium oceny danej transformacji, np. w algorytmach kompresji obrazu. Zagadnienia te należą już do kanonów przetwarzania obrazów i są obszernie omawiane w wielu pracach [11,35,37,40]. Nie ma więc potrzeby, aby powtarzać te wywody. Niektóre zagadnienia cyfrowego przetwarzania obrazów można jednak przedstawić inaczej, uwzględniając przytoczone w poprzednich rozdziałach wywody teoretyczne. Pozwala to na głębsze zrozumienie procesów zachodzących w dyskretnych przekształceniach 2D i jest warunkiem powodzenia własnych eksperymentów.

Ogólna postać dwuwymiarowej transformacji Fouriera $M \times N$ wymiarowego obrazu $\boldsymbol{O}$ w różnych bazach ma postać:

$$\boldsymbol{S} = \alpha \cdot \boldsymbol{J}_a \cdot \boldsymbol{O} \cdot \boldsymbol{J}_b^T. \tag{6.1}$$

Transformacja odwrotna ma postać:

$$\boldsymbol{O} = \beta \cdot \boldsymbol{J}_a^T \cdot \boldsymbol{S} \cdot \boldsymbol{J}_b, \tag{6.2}$$

gdzie $\boldsymbol{J}$ jest macierzą jądrową odpowiedniego przekształcenia, $\boldsymbol{S}$ jest macierzą współczynników widmowych, $\alpha \cdot \beta = 1/(M \cdot N)$.

Porównanie widm można przeprowadzić na podstawie użytego wielokrotnie obrazu wzorcowego, który najlepiej uwidoczni różnice poszczególnych transformat i pozwoli ocenić przydatność poszczególnych widm w przetwarzaniu obrazów. Substytutem takiego obrazu może być macierz kwadratowa $\boldsymbol{O}$ o rozmiarach $N \times N$, $N = 2^n$, wypełniona następująco:

$$\boldsymbol{O} = \begin{bmatrix} N & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{6.3}$$

Niech obraz $\boldsymbol{O}$ jest macierzą kwadratową o rozmiarze 8×8, wtedy $\boldsymbol{J} = \boldsymbol{J}_a = \boldsymbol{J}_b$. Niech $\alpha = 1$ i $\beta = 1/N^2$.

Omawiane już macierze jądrowe wybranych transformacji dla $N = 8$ przedstawiają się następująco.

Macierz transformacji DST:

$$\boldsymbol{J} = [j_{k,n}]_{8\times 8} = \sin\left(\frac{\pi kn}{N+1}\right) =$$

$$= \begin{bmatrix} 0.3420 & 0.6428 & 0.8660 & 0.9848 & 0.9848 & 0.8660 & 0.6428 & 0.3420 \\ 0.6428 & 0.9848 & 0.8660 & 0.3420 & -0.3420 & -0.8660 & -0.9848 & -0.6428 \\ 0.8660 & 0.8660 & 0.0000 & -0.8660 & -0.8660 & 0.0000 & 0.8660 & 0.8660 \\ 0.9848 & 0.3420 & -0.8660 & -0.6428 & 0.6428 & 0.8660 & -0.3420 & -0.9848 \\ 0.9848 & -0.3420 & -0.8660 & 0.6428 & 0.6428 & -0.8660 & -0.3420 & 0.9848 \\ 0.8660 & -0.8660 & 0.0000 & 0.8660 & -0.8660 & 0.0000 & 0.8660 & -0.8660 \\ 0.6428 & -0.9848 & 0.8660 & -0.3420 & -0.3420 & 0.8660 & -0.9848 & 0.6428 \\ 0.3420 & -0.6428 & 0.8660 & -0.9848 & 0.9848 & -0.8660 & 0.6428 & -0.3420 \end{bmatrix}.$$

Pozostałe macierze jądrowe budowane są podobnie.

Macierz transformacji DCT:

$$\boldsymbol{J} = [j_{k,n}]_{8\times 8} = c(k)\cos\left(\frac{\pi k(2n+1)}{2N}\right) =$$

$$= \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}.$$

Macierz transformacji Hartleya:

$$\boldsymbol{J} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \sqrt{2} & 1 & 0 & -1 & -\sqrt{2} & -1 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & -1 & \sqrt{2} & -1 & 0 & 1 & -\sqrt{2} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\sqrt{2} & 1 & 0 & -1 & \sqrt{2} & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & -1 & -\sqrt{2} & -1 & 0 & 1 & \sqrt{2} \end{bmatrix}. \tag{6.4}$$

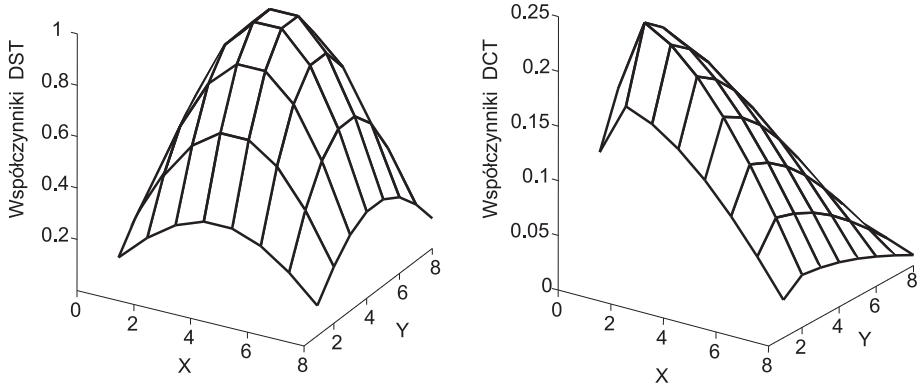
Macierz transformacji Haara:

$$\boldsymbol{J} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}. \tag{6.5}$$

Macierz transformacji Walsha:

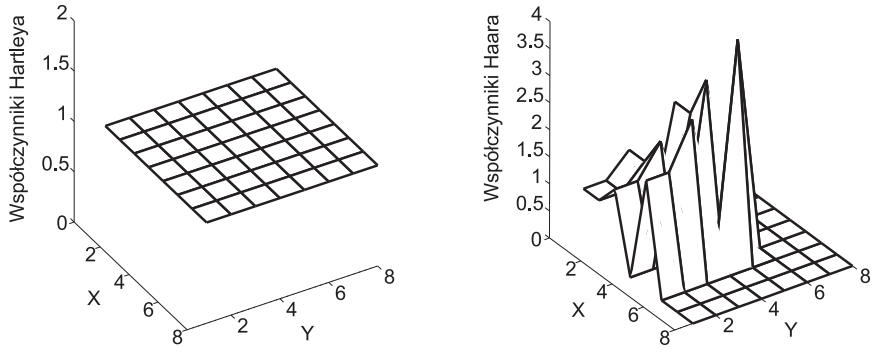
$$\boldsymbol{J} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}. \tag{6.6}$$

Widmo obrazu $\boldsymbol{O}$ dla różnych macierzy jądrowych $\boldsymbol{J}$ można przedstawić w postaci wykresów macierzy. W Matlabie służy do tego celu funkcja $\mathbf{mesh}(\boldsymbol{S})$, gdzie $\boldsymbol{S}$ jest odpowiednią macierzą współczynników widmowych. Wykresy niektórych widm obrazu $\boldsymbol{O}$ przedstawione zostały na rys. 17 i rys. 18. Na wykresach można zaobserwować charakterystyczne rozkłady współczynników widmowych. Wyjaśnia to, dlaczego zastosowanie różnych transformat silnie zależy od charakteru rozkładu współczynników w poszczególnych partiach macierzy widma. Transformacje DCT oraz DST doskonale nadają się do kompresji obrazów, jak bowiem łatwo zauważyć, współczynniki widmowe charakterystycznie (nierównomiernie) się rozkładają [28,37]. Mają wyraźnie zaznaczone obszary, gdzie występują duże wartości współczynników widmowych, oraz obszary wartości silnie malejących. Współczynniki o małych wartościach nie dostarczają istotnych informacji o obrazie, można je zatem pomijać, przypisując im np. wartość zero.



Rys. 17. Widmo DST oraz DCT obrazu $\boldsymbol{O}$

Takiego zjawiska nie obserwuje się w przypadku widma Walsha i Hartleya, dlatego też te transformacje nie znalazły zastosowania w kompresji obrazów. Charakter transformacji Haara jest inny i można ją stosować w dedykowanej kompresji, gdzie można wpływać na zmiany obrazu w określonych kierunkach – oddziaływać na pionowe, poziome lub ukośne elementy obrazu.



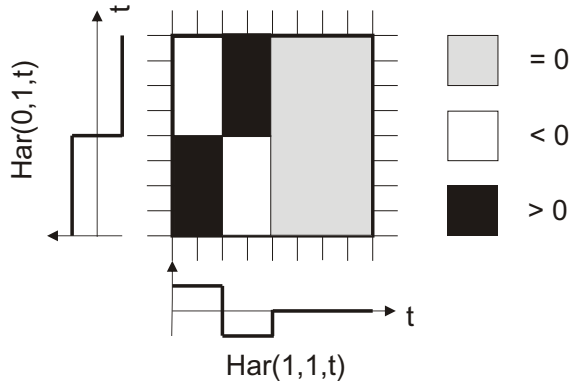
Rys. 18. Widmo Hartleya oraz Haara obrazu $\boldsymbol{O}$. Widmo Walsha dla tego przypadku jest takie samo, jak widmo Hartleya

Zarówno dwuwymiarowy obraz-sygnał $\boldsymbol{O}$, jak i jego widmo $\boldsymbol{S}$ mogą być macierzami o rozmiarach $M \times N$. Macierz $\boldsymbol{J}$ można zapisać osobno – wierszami i kolumnami. Otrzymamy wtedy macierze wierszowe i kolumnowe $\boldsymbol{J}_a$ oraz $\boldsymbol{J}_b$. Macierz $\boldsymbol{J}_a$ ma wymiar $M \times M$, a macierz $\boldsymbol{J}_b$ – wymiar $N \times N$. Transformata (6.1) obrazu $\boldsymbol{O}$ jest liniową kombinacją dwuwymiarowych funkcji bazowych:

$$\begin{aligned} \boldsymbol{S} &= \left[\boldsymbol{j}_0, ..., \boldsymbol{j}_{M-1}\right] \cdot \begin{bmatrix} o(0,0) & \cdots & o(0,N-1) \\ \vdots & \ddots & \vdots \\ o(M-1,0) & \cdots & o(M-1,N-1) \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{j}_0^T \\ \vdots \\ \boldsymbol{j}_{N-1}^T \end{bmatrix} \\ &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} o[m,n] \cdot \boldsymbol{j}_m \boldsymbol{j}_n^T, \end{aligned} \tag{6.7}$$

gdzie $\boldsymbol{j}_m$ i $\boldsymbol{j}_n$ reprezentują odpowiednio m-tą oraz n-tą jednowymiarową funkcję bazową odpowiedniej macierzy jądrowej $\boldsymbol{J}_a$ i $\boldsymbol{J}_b$. Jeśli przyjąć, że $M = N = 8$ i zaniedbać wartości funkcji bazowych, to zasadę mnożenia wektorów $\boldsymbol{j}_m \boldsymbol{j}_n^T$ we wzorze (6.7) przedstawić można graficznie. Na potrzeby demonstracji jako funkcje bazowe zastosowano funkcje Haara $haar()$.

Mogą to być jednak dowolne funkcje bazowe. Dla obrazu $\boldsymbol{O}$ o rozmiarach 8×8 można zbudować 64 grupy iloczynów odpowiednich funkcji bazowych, konstruowanych według schematu przedstawionego na rys. 19. Każda z grup jest kombinacją liniową odpowiednich funkcji bazowych.



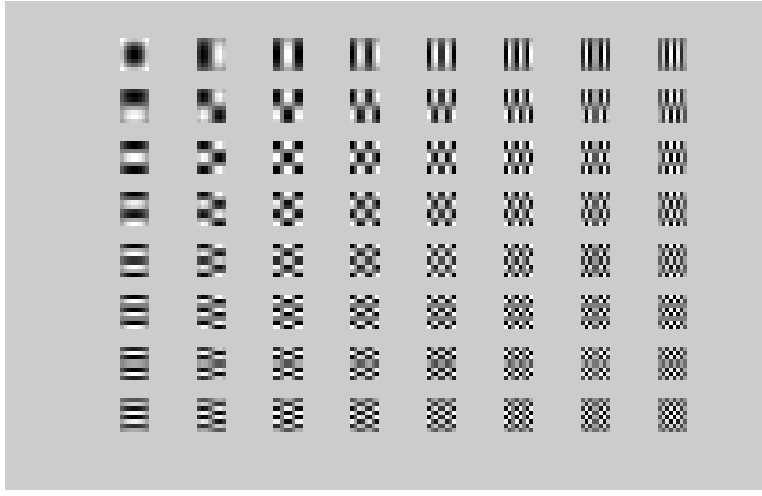
Rys. 19. Graficzna interpretacja iloczynu wektorów bazowych w dwuwymiarowej transformacji Haara

Obrazy iloczynów funkcji bazowych, a w konsekwencji obrazy różnych transformacji, można przedstawić, korzystując z prostego programu. W zależności od zastosowanych funkcji bazowych ich iloczyny mają różne postaci, a każdą grupę iloczynów przedstawić można graficznie.

Program 6.1. *Program graficznej reprezentacji iloczynów funkcji bazowych transformacji dwuwymiarowych. Poniżej przedstawiono program operujący na funkcjach Walsha–Hadamarda. W przypadku innych funkcji bazowych należy w programie podać odpowiednią definicję funkcji, np. macierze DST, DCT itp.*

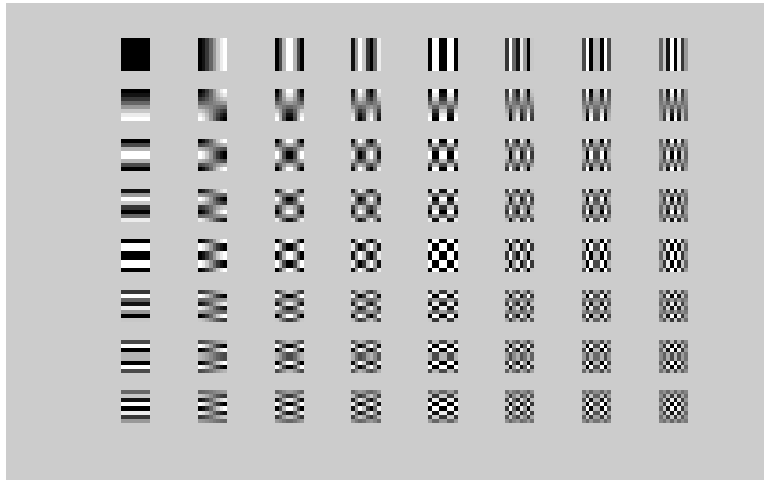
```
N=8;              % Wymiar macierzy jadrowej
S=hadamard(N);    % Beda budowane wykresy funkcji Walsha-Hadamarda
s=1;              % Ustawienie wartosci poczatkowej licznika
W=zeros(N,N);     % Macierz przechowujaca grupe iloczynow
for w=1:N         % funkcji bazowych
```

```
    for k=1:N
        for c=1:N
            W(c,:)=S(w,c)*S(k,:);
            subplot(N,N,s);     % Miejsce obrazu na panelu
            imshow(-1*W,[]);    % Obraz iloczynu funkcji bazowych
        end
        s=s+1;                  % Inkrementacja licznika petli
    end
end
```

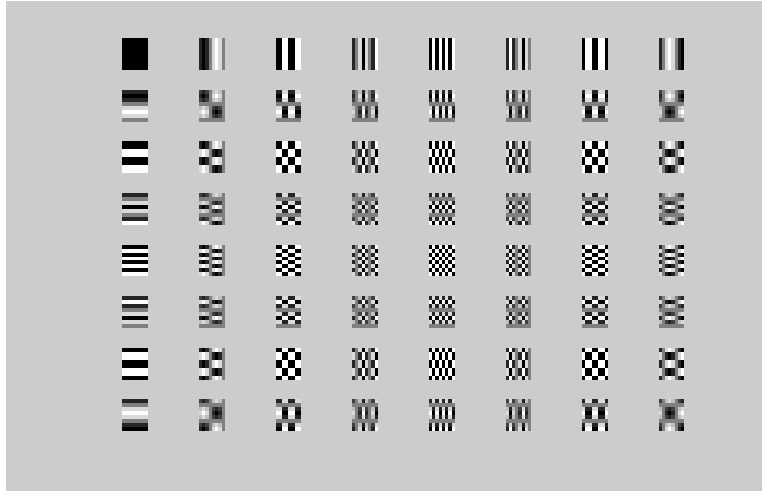


Rys. 20. Iloczyny funkcji bazowych dwuwymiarowej transformacji sinusowej (DST)

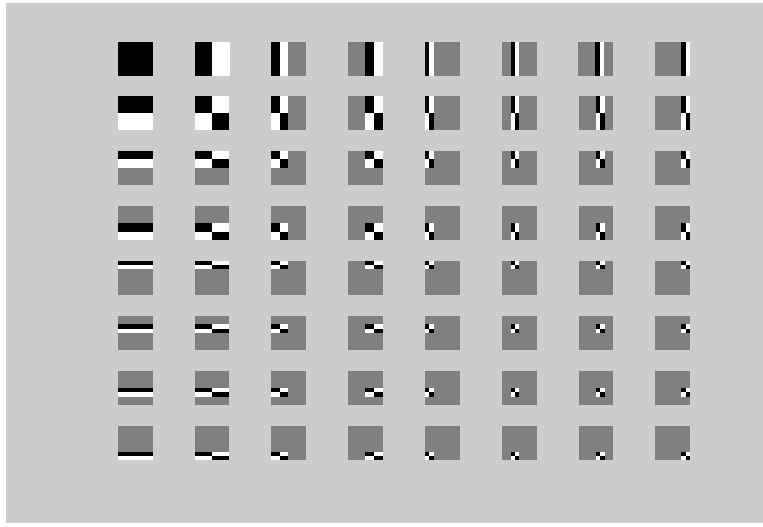
W podobny sposób postąpić można w przypadku innych transformacji, dla których znane są ich macierze jądrowe. Dla transformacji dwuwymiarowych otrzymamy wtedy różne obrazy iloczynów funkcji bazowych w poszczególnych grupach. Ilustrują to kolejne rysunki przedstawione poniżej. Rysunki, przeniesione z Matlaba, są efektem działania Programu 6.1, gdzie wymianie podlegają wyłącznie definicje macierzy jądrowych.



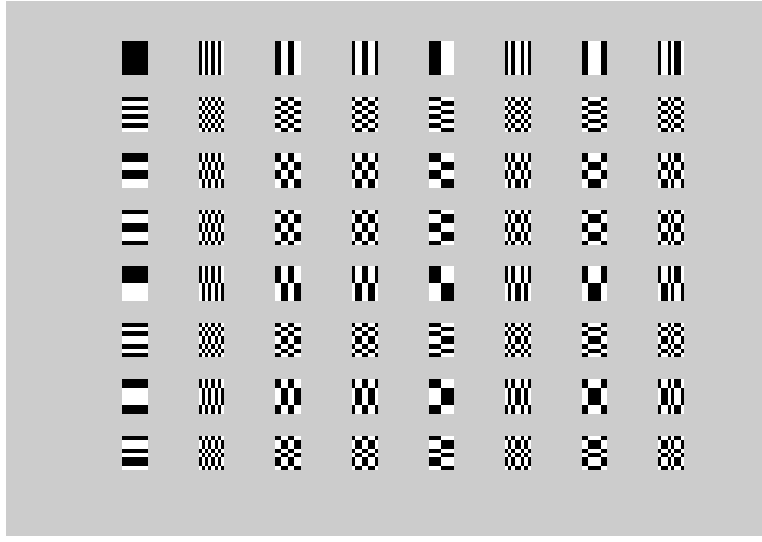
Rys. 21. Iloczyny funkcji bazowych dwuwymiarowej transformacji kosinusowej (DCT)



Rys. 22. Iloczyny funkcji bazowych dwuwymiarowej transformacji Hartleya



Rys. 23. Iloczyny funkcji bazowych dwuwymiarowej transformacji Haara



Rys. 24. Iloczyny funkcji bazowych dwuwymiarowej transformacji Walsha–Hadamarda

Analiza iloczynów funkcji bazowych Haara (rys. 23) wskazuje na specyficzne uporządkowanie tych funkcji, z wyraźnym podziałem utworzonych w wyniku tych mnożeń obszarów, w których wyróżnić można poziome, pionowe i diagonalne fragmenty. Takie właściwości rozkładu funkcji bazowych w przestrzeni dwuwymiarowej trudno zaobserwować w innych transformacjach. Pokazane obszary są ukrytym, trudnym do zaobserwowania elementem procesu przetwarzania danych. Znając te obszary, można wpływać na przebieg procesu przetwarzania i wykorzystać do dedykowanego redukowania danych z rzeczywistego obrazu cyfrowego. Przebieg procesu przetwarzania obrazu pierwotnego będzie kształtowany za pomocą masek Haara, które w praktyce powodują wyzerowanie wybranych fragmentów widma analizowanego obrazu. Na podstawie tak spreparowanego widma będzie odtwarzany obraz pierwotny. Będzie to oczywiście obraz zniekształcony, a charakter tego zniekształcenia zależy od wybranej maski Haara. Oddziaływanie masek Haara na rzeczywiste obrazy przedstawione zostało poniżej.

W demonstrowanych przykładach wczytywane obrazy bitmapowe mają rozmiar 256×256 i są zapisywane w 8-bitowej głębi kolorów. Przygotowano 4 maski: M1, M2, M3 oraz M4, które w operacji bezkontekstowej są nakładane na widmo oryginalnego obrazu. Maski modyfikują widmo i powodują wyzerowanie 25% współczynników widmowych, które ulokowane są zawsze w jednej z czterech podmacierzy $N/2 \times N/2$ macierzy $N \times N$ obrazu widma. Grupy zerowych pozycji oraz grupy współczynników widmowych o relatywnie małej wartości można wtedy efektywnie kodować. Miejsca występowania obszarów wyzerowanych (elementy o wartości równej 0) będą zobrazowane na rysunkach. Na potrzeby wizualizacji danych obraz oraz jego widmo Haara są najpierw odpowiednio normalizowane. Macierz, w której przechowywane jest widmo, jest w odpowiednich częściach zerowana, a następnie na podstawie tak spreparowanego widma obraz jest odtwarzany. Wyniki tych operacji można przedstawić graficznie – w postaci obrazów. Ponieważ oryginalne widmo obrazu było zmieniane za pomocą masek M, odtworzony obraz będzie zdeformowany. Poziom tej deformacji można jednak programować za pomocą masek.

Odpowiednio przygotowany program w Matlabie pozwala na wykonywanie opisanych operacji w sposób automatyczny. Wykorzystując współczynnik PSNR (ang. *Peak Signal-to-Noise Ratio*), można ocenić poziom podobieństwa obrazu pierwotnego (ob) do obrazu odtworzonego (ob^*) [30]. Należy wziąć pod uwagę, że współczynnik PSNR przyjmuje wartości w skali logarytmicznej – im mniejsza jest jego wartość, tym mniejsze podobieństwo między porównywanymi obrazami. Dla obrazów o rozmiarze 256×256 zapisanych w skali szarości współczynnik PSNR wyznaczyć można z uproszczonej zależności:

$$PSNR = 20 \log_{10} \frac{255}{\sum_{m=0}^{255} \sum_{n=0}^{255} [ob(m,n) - ob^*(m,n)]^2} \ , \tag{6.8}$$

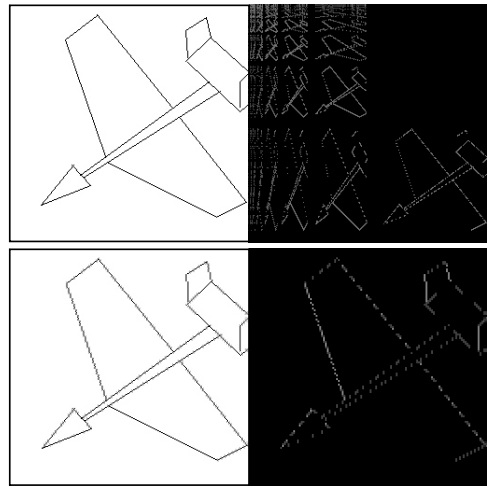
gdzie m, n są pikselowymi współrzędnymi analizowanego obrazu bitmapowego.

Program 6.2. *Interpretacja widma Haara i znajdowanie różnic między obrazem oryginalnym a odtworzonym po zastosowaniu różnych masek Haara. Program nie zawiera zabezpieczeń dotyczących prawidłowego rozmiaru obrazu wejściowego. Obraz wejściowy powinien być zawsze zapisany w formacie* $256 \times 256 \times 8$.

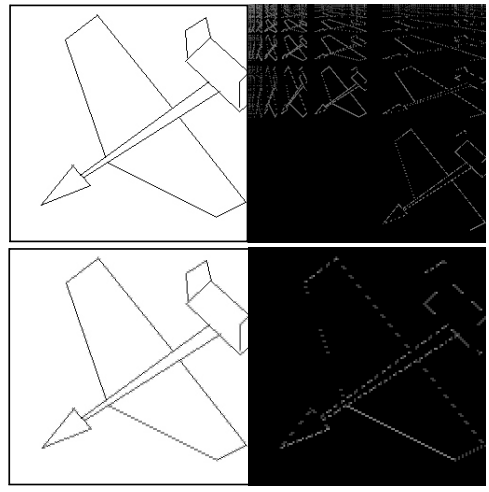
```
% Wybor obrazu z dysku komputera
[FiN, PNa]=uigetfile('*.BMP', 'Wskaz plik BMP');
% Zaladowanie obrazu PNa w formacie BMP
Ob=imread([PNa, FiN]);
% Normalizacja obrazu umieszczonego w macierzy Ob
Ob=double(Ob)/255;
% Obraz bitmapowy znormalizowany
Ob=imadjust(Ob,stretchlim(Ob), [0 1]);
Hr=[1];
N=256;
p=ceil(log2(N));            % Wyznaczenie liczby iteracji
for m=1:p                   % Tworzenie macierzy jadrowej Haara
    A=kron(Hr,[1 1]);       % Pierwszy wiersz macierzy blokowej Hr
```

```
    B = kron(2^((m-1)/2)*eye(2^(m-1)),[1 -1]);
    Hr = [A;B];              % Kompletna macierz Haara
end
g=Hr*Ob*Hr';                 % Widmo Haara obrazu oryginalnego Ob
m=max(max(g));               % Najwieksza wartosc wsp. widmowego
c=real(1.0/(log(1+m)));
ng=real(c*log(1+g));   % Widmo znormalizowane
M=ones(256,256); M(1:128,1:128)=0; M1=M;        % Maska M1
M=ones(256,256); M(1:128,128:256)=0; M2=M;      % Maska M2
M=ones(256,256); M(128:256,1:128)=0; M3=M;      % Maska M3
M=ones(256,256); M(128:256,128:256)=0; M4=M;    % Maska M4
maska=M2;              % Wybor maski. Tutaj wybrano maske M2
g2=ng.*maska;          % Widmo znormalizowane po nalozeniu maski M2
g=g.*maska;            % Widmo oryginalne po nalozeniu maski M2
G2=(1/256^2)*Hr'*g*Hr;   % Odtworzenie obrazu ze
                         % zmodyfikowanego widma g
figure(1)
imshow([Ob, g2]);      % Wyswietlenie obrazu oryginalnego oraz
figure(2)              % znormalizowanego widma po nalozeniu maski
imshow([G2, abs(Ob-G2)]);      % Wyswietlenie obrazu odtworzonego
                         % na podstawie widma oraz roznicy obrazow
title('Roznica obrazow');
diff=(Ob*256-G2.*256).^2;
s=sum(sum(diff));
MSE=s/(256^2);       % Blad sredniokwadratowy MSE roznicy obrazow
psnr=10*log(255.0/sqrt(MSE));        % Wspolczynnik sygnal-szum
```

Seria obrazów (samolot.bmp) odtworzonych z odpowiednio przygotowanego (wyzerowanego w odpowiednich miejscach) widma została przedstawiona poniżej. Na rysunkach zaznaczono również podobieństwa obrazów – oryginalnego i odtworzonego – wyrażone decybelową skalą współczynnika PSNR.

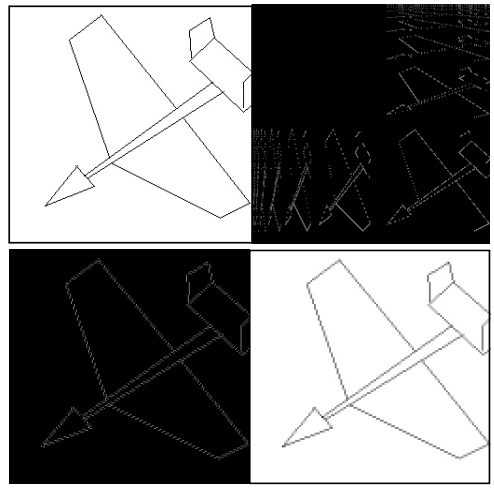


Rys. 25. Obraz oryginalny (u góry po lewej) oraz kolejno: jego widmo Haara z miejscem, gdzie dokonano wyzerowania współczynników maską M2, obraz odtworzony i elementy utracone w wyniku zerowania widma. PSNR=58 dB



Rys. 26. Obraz oryginalny (u góry po lewej) oraz kolejno: jego widmo Haara z miejscem, gdzie dokonano wyzerowania współczynników maską M3, obraz odtworzony i elementy utracone w wyniku zerowania widma. PSNR=57 dB

Należy zwrócić uwagę, że nie powinno się zerować współczynników widmowych w dowolnym miejscu i należy przestrzegać zasad, które wynikają z obserwacji rys. 23. Największa energia widma skupiona jest w pierwszej ćwiartce macierzy widma. Zastosowanie maski M1 całkowicie deformuje obraz odtwarzany. Ten przypadek pokazano na rys. 27.



Rys. 27. Obraz oryginalny (u góry po lewej) oraz kolejno: jego widmo Haara z miejscem, gdzie dokonano wyzerowania współczynników maską M1, obraz odtworzony i elementy utracone w wyniku zerowania widma. PSNR=0 dB

6.2. Transformacja z falką Haara

W latach 80. ubiegłego wieku zauważono, że zbiór funkcji Haara można konstruować na bazie jednej funkcji "matki" przez odpowiednie jej przesunięcia i skalowanie. Niech $\psi : \boldsymbol{R} \to \boldsymbol{R}$ oraz:

$$\psi(t) = \begin{cases} +1 & \text{dla } t \in [0, \frac{1}{2}) \\ -1 & \text{dla } t \in [\frac{1}{2}, 1) \\ 0 & \text{dla innych } t \end{cases} . \tag{6.9}$$

Niech:

$$\psi_i^j(t) = \sqrt{2^j}\psi(2^j t - i), \quad i = 0, 1, ..., 2^j - 1, \quad j = 0, 1, ..., \log_2 N - 1. \quad (6.10)$$

Stała $\sqrt{2^j}$ sprawia, że iloczyn skalarny $< \psi_i^j, \psi_i^j > = 1$, $\psi_i^j \in L^2(\mathbf{R})$.
Niech $N = 8$, a funkcjami ψ będą funkcje Haara.
Otrzymujemy wtedy: $\psi_0^0(t) = haar(1,t)$, $\psi_0^1(t) = haar(2,t)$, $\psi_1^1(t) = haar(3,t)$, $\psi_0^2(t) = haar(4,t)$, $\psi_1^2(t) = haar(5,t)$, $\psi_2^2(t) = haar(6,t)$, $\psi_3^2(t) = haar(7,t)$.
Jest to więc system funkcji, które szczegółowo omawiane były już w poprzednich rozdziałach. Dla dowolnego N każda funkcja $\psi_i^j(t)$ powstaje z odpowiedniej funkcji Haara $\psi_i^j(t) = haar(2^j + i, t)$. System funkcji $\psi_i^j(t)$ jest systemem funkcji ortogonalnych. System ten rozpina przestrzeń wektorową $W^j = span\{\psi_i^j\}_{i=0,...,2^j-1}$. Zbiór liniowo niezależnych funkcji $\{\psi_i^j(t)\}_{i=0,1,...,2^j-1}$ rozpinających W^j nazywamy falkami. Dyskretna transformata falkowa opisywana jest jako rzutowanie elementów przestrzeni sygnału na jej bazę. System takich funkcji jest podstawą analizy wielorozdzielczej [5], [38].

Niech $\phi : \boldsymbol{R} \to \boldsymbol{R}$. Dla omawianych funkcji Haara funkcją skalującą jest funkcja:

$$\phi(t) = \begin{cases} 1 \text{ dla } t \in [0,1) \\ 0 \text{ dla } t \notin [0,1) \end{cases}, \quad (6.11)$$

na podstawie której można utworzyć rodzinę funkcji:

$$\phi_i^j(t) = \sqrt{2^j}\phi(2^j t - i), \quad i = 0, 1, ..., 2^j - 1, \quad j = 0, 1, ..., \log_2 N - 1. \quad (6.12)$$

Podobnie jak poprzednio, stała normalizująca $\sqrt{2^j}$ sprawia, że $< \phi_i^j, \phi_i^j > = 1$, $\phi_i^j \in L^2(\boldsymbol{R})$. Indeks j funkcji ϕ_i^j nazywany jest współczynnikiem dylatacji, a indeks i – współczynnikiem translacji. Zbiór funkcji ϕ_i^j rozpina przestrzeń wektorową $V^j = span\{\phi_i^j\}_{i=0,...,2^j-1}$. Funkcje ϕ_i^j przestrzeni V^j są więc funkcjami skalującymi. Dla dowolnego j ortogonalnym dopełnieniem przestrzeni V^j w przestrzeni V^{j-1} jest przestrzeń funkcji ψ_i^j. Oznaczając tę przestrzeń jako W^j, otrzymujemy:

$$V^j = V^{j-1} \oplus W^{j-1}, \quad (6.13)$$

co oznacza, że przestrzeń wektorowa W^j może być traktowana jako ortogonalne dopełnienie przestrzeni V^j w przestrzeni V^{j+1}.

Innymi słowy, przestrzeń W^j jest przestrzenią wszystkich funkcji w V^{j+1}, które są ortogonalne do wszystkich funkcji w V^j. Z tego powodu funkcje bazowe $\psi_i^j \in W^j$ razem z funkcjami bazowymi $\phi_i^j \in V^j$ formują bazę w V^{j+1} i każda funkcja bazowa ψ_i^j jest ortogonalna do każdej funkcji ϕ_i^j. Oznaczmy klasyczną macierz Haara (5.190) jako $\boldsymbol{H}(n)$.
Dla tej macierzy, można wyznaczyć jej macierze dekompozycji, zgodnie z zasadami wielorozdzielczej syntezy Mallata [20]:

Krok 1
Ponieważ $V^n = V^{n-1} \oplus W^{n-1}$, macierz $\boldsymbol{M}_1$ ma formę:

$$\boldsymbol{M}_1 = [\phi^{n-1}_{j=0,\ldots,2^{n-1}-1} \subset V^{n-1}, \psi^{n-1}_{j=0,\ldots,2^{n-1}-1} \subset W^{n-1}]^T. \tag{6.14}$$

Krok 2
Ponieważ $V^{n-1} = V^{n-2} \oplus W^{n-2} \oplus W^{n-1}$, macierz $\boldsymbol{M}_2$ ma następującą konstrukcję:

$$\boldsymbol{M}_2 = [\phi^{n-2}_{j=0,\ldots,2^{n-2}-1} \subset V^{n-2}, \psi^{n-2}_{j=0,\ldots,2^{n-2}-1} \subset W^{n-2}, \psi^{n-1}_{j=0\ldots,2^{n-1}-1} \subset W^{n-1}]^T. \tag{6.15}$$

Krok n
Po n-tym kroku obliczeń mamy: $V^1 = V^0 \oplus W^0 \oplus W^1 \oplus W^2 \oplus \ldots \oplus W^{n-1}$. Macierz $\boldsymbol{M}_n$ jest zbudowana następująco:

$$\boldsymbol{M}_n = [\phi^0_0 \subset V^0, \psi^0_0 \subset W^0, \psi^1_{j=0,1} \subset W^1, \psi^2_{j=0,\ldots,3} \subset W^2, \ldots, \psi^{n-1}_{j=0,\ldots,2^{n-1}-1} \subset W^{n-1}]^T \tag{6.16}$$

Powyższe wywody mają zastosowania praktyczne – z ich pomocą można ustalać poziom stratnej kompresji obrazu, a tym samym jego jakość, co zobrazować można odpowiednimi przykładami.

Przykład 6.1. *Niech $n = 3$, wtedy: $V^3 = V^2 \oplus W^2$ oraz:*

$$\boldsymbol{M}_1 = [\phi_0^2, \phi_1^2, \phi_2^2, \phi_3^2, \psi_0^2, \psi_1^2, \psi_2^2, \psi_3^2]^T = \begin{bmatrix} 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}.$$

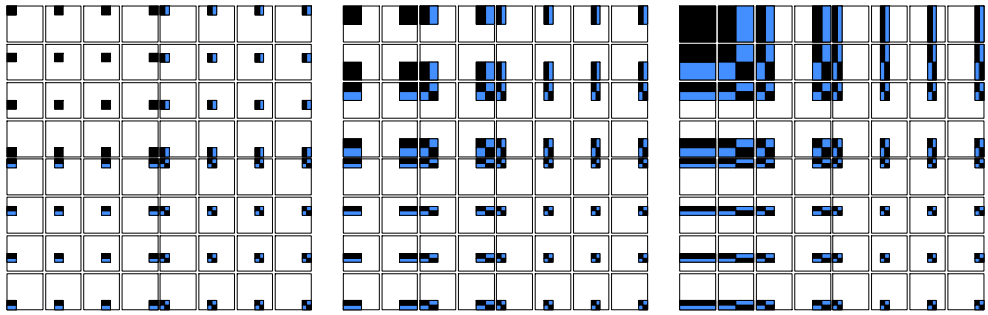
$V^2 = V^1 \oplus W^1 \oplus W^2$, *zatem*:

$$\boldsymbol{M}_2 = [\phi_0^1, \phi_1^1, \psi_0^1, \psi_1^1, \psi_{j=0\ldots,3}^2 \subset W^2]^T = \\ = \begin{bmatrix} \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}.$$

$V^1 = V^0 \oplus W^0 \oplus W^1 \oplus W^2$, *a więc*:

$$\boldsymbol{M}_3 = [\phi_0^0, \psi_0^0, \psi_{j=0,1}^1 \subset W^1, \psi_{j=0,\ldots 3}^2 \subset W^2]^T = \\ = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}.$$

Ostatni poziom dekompozycji oznacza, że $\boldsymbol{M}_3 = \boldsymbol{H}(3)$. Jeśli każdą ortogonalną macierz $\boldsymbol{M}_i$ $i = 1, 2, 3$ pomnożyć przez czynnik skalujący $1/\sqrt{2}$, to obliczenia prowadzą do wyniku zgodnego z klasycznym algorytmem dekompozycji Mallata [25]. Stosując identyczne reguły oznaczeń jak na rys. 19, poszczególne poziomy dekompozycji, dla $N = 8$, można przedstawić graficznie. Są to ekstraktory cech przywoływanego już w poprzednim rozdziale obrazu $\boldsymbol{O}$ dla różnych poziomów dekompozycji. Opisana zasada wielopoziomowego dekomponowania



Rys. 28. Ekstraktory Haara: pierwszy, drugi oraz trzeci poziom dekompozycji

macierzy Haara może być przedstawiona w tej samej przestrzeni w inny sposób. Bazując na właściwościach funkcji Haara i wiedząc, że $W^j = span\{\varphi_i^j\}_{i=0,\ldots,2^j-1}$ oraz $V^j = span\{\phi_i^j\}_{i=0,\ldots,2^j-1}$, można te funkcje opisać jako liniową kombinację funkcji z przestrzeni W oraz V. W przypadku falki Haara mamy:

$$\phi(t) = h(0)\sqrt{2}\phi(2t) + h(1)\sqrt{2}\phi(2t-1), \tag{6.17}$$

$$\psi(t) = g(0)\sqrt{2}\phi(2t) + g(1)\sqrt{2}\phi(2t-1), \tag{6.18}$$

gdzie: $\{h(0), h(1)\}$ i $\{g(0), g(1)\}$ są dolno- i górnoprzepustowym filtrem.

W naszym przypadku współczynniki $h(k)$ i $g(k)$, $k = 0, 1$ są określone następująco [5]:

$$h(0) = \frac{1}{\sqrt{2}},\ h(1) = \frac{1}{\sqrt{2}},\ g(0) = \frac{1}{\sqrt{2}},\ g(1) = -\frac{1}{\sqrt{2}}. \tag{6.19}$$

Podobnie jak poprzednio, oznaczmy literą $\boldsymbol{O}$ macierz obrazu oraz zdefiniujmy następujące operatory:

$$\boldsymbol{A}(i) = \frac{1}{\sqrt{2}}\boldsymbol{O}(2i) + \frac{1}{\sqrt{2}}\boldsymbol{O}(2i+1), \tag{6.20}$$

$$\boldsymbol{D}(i) = \frac{1}{\sqrt{2}}\boldsymbol{O}(2i) - \frac{1}{\sqrt{2}}\boldsymbol{O}(2i+1), \tag{6.21}$$

gdzie:

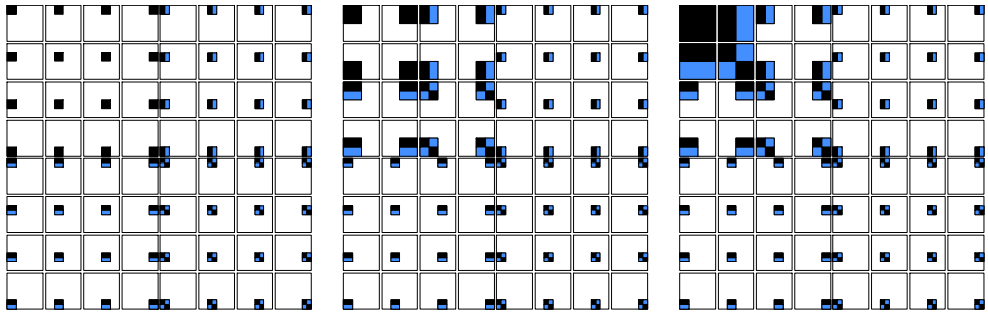
$\boldsymbol{O}(i)$ – wektor N elementowy, zawierający wiersz lub kolumnę macierzy $\boldsymbol{O}$ (obrazu), gdzie $i \in \{0, 1, ..., \frac{N}{2} - 1\}$,

$\boldsymbol{A}(i)$ – wektor $N/2$ elementowy, zawierający współczynniki aproksymacji,

$\boldsymbol{D}(i)$ – wektor $N/2$ elementowy zawierający współczynniki detali.

Na pierwszym poziomie dekompozycji obrazu $\boldsymbol{O}$ jego widmo przechowywane będzie w macierzy $\boldsymbol{S}_1$. Ten poziom osiągany jest dzięki zastosowaniu operatorów (6.20) i (6.21) najpierw do wszystkich kolumn, a potem do wszystkich wierszy macierzy. Podobna analiza może być przeprowadzona dla drugiego poziomu dekompozycji obrazu. Widmo drugiego poziomu dekompozycji przechowywane będzie w macierzy $\boldsymbol{S}_2$. Operacje dekompozycji są wtedy ograniczone do lewej, górnej podmacierzy o rozmiarach $\frac{N}{2} \times \frac{N}{2}$ macierzy $\boldsymbol{S}_1$. Dla k-tego poziomu dekompozycji widmo przechowywane jest w macierzy $\boldsymbol{S}_k$, a operacje dekompozycji przeprowadzane są w lewej, górnej podmacierzy o rozmiarach $\frac{N}{2^{k-1}} \times \frac{N}{2^{k-1}}$ macierzy $\boldsymbol{S}_{k-1}$, gdzie $k \in \{1, ..., \log_2 N\}$.

Można zauważyć, że użycie operatorów (6.20) i (6.21) dla obrazu $\boldsymbol{O}$ daje taki sam wynik, jak wykonanie operacji macierzowej $\boldsymbol{S}_1 = \frac{1}{8}\boldsymbol{M}_1 \cdot \boldsymbol{O} \cdot \boldsymbol{M}_1^T$, gdzie macierz $\boldsymbol{M}_1$ jest macierzą z przykładu (6.1). Macierz $\boldsymbol{S}_1$ może być traktowana jako ekstraktor cech obrazu $\boldsymbol{O}$ na pierwszym poziomie dekompozycji falkowej, podobnie jak dekompozycja za pomocą funkcji Haara. Drugi i kolejne poziomy dekompozycji falkowej są już jednak inne, gdyż tylko część macierzy jest transformowana. Tak więc ekstraktory falkowe cech obrazu są inne niż ekstraktory Haara. Trójpoziomowa dekompozycja falkowa dla obrazów o rozmiarach $N \times N$, $N = 8$ została przedstawiona na rys. 29 [25].



Rys. 29. Ekstraktory falkowe: pierwszy, drugi oraz trzeci poziom dekompozycji

Rozważania dotyczące klasycznych funkcji Haara są również ważne dla falek Haara, z wyjątkiem rozkładu elementów ekstraktorów. Omawiane już filtry $h(k)$, $k = 0, 1$ mogą być opisane iloczynami skalarnymi:

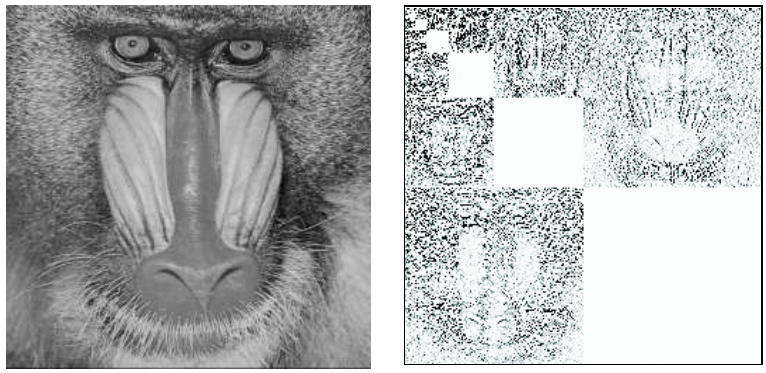
$$h(k) =< \phi(t), \sqrt{2}\phi(2t - k) >, \tag{6.22}$$

$$g(k) =< \psi(t), \sqrt{2}\phi(2t - k) > . \tag{6.23}$$

Prezentowane powyżej sposoby dekompozycji mają liczne zastosowania praktyczne [5, 30, 37]. Pomiędzy wielopoziomową dekompozycją macierzy Haara a falkami Haara zachodzą zależności, które można przedstawić graficznie. Dla dowolnego obrazu $\boldsymbol{O}$ jego widmo Haara i widmo falkowe są proporcjonalne w obszarach zaznaczonych białymi kwadratami. Rysunek 30 prezentuje obraz oryginalny (po lewej) oraz obraz widma Haara z nałożonymi obszarami, wewnątrz których występują wartości proporcjonalne do wartości widma falkowego. Jeśli na danym poziomie dekompozycji macierz współczynników widmowych ma wymiar $2^n \times 2^n$, $n = 0, 1, ..., \log_2 N - 1$, to liczba współczynników w zaznaczonych na rys. 30 obszarach wynosi $\sum_{i=0}^{n-1} 2^i \cdot 2^i$.

Oznacza to, że ok. 1/3 współczynników widmowych w macierzy Haara jest proporcjonalna do współczynników macierzy falkowej, gdyż [25]:

$$\frac{\sum\limits_{i=0}^{n-1} 2^i \cdot 2^i}{2^n \cdot 2^n} \approx \frac{1}{3}. \tag{6.24}$$



Rys. 30. Podobieństwa widma Haara i widma falkowego (białe obszary)

6.3. Widmowa analiza binarnych funkcji boolowskich

Znajomość widma funkcji boolowskiej i rozkład tego widma pozwalają na rozpoznawanie pewnych klas takich funkcji. Ma to na przykład duże znaczenie dla boolowskich funkcji liniowych lub tzw. funkcji giętych (ang. *bent functions*). W algorytmach kryptograficznych żądamy bowiem, aby stosowane tam funkcje boolowskie były nieliniowe ze względu na możliwości przełamania szyfru. Jednakże stosowanie funkcji liniowych w rozwiązaniach sprzętowych jest ze wszech miar pożądane, ze względu na korzystną złożoność realizacyjną układu – wykorzystuje się wtedy mniejszą liczbę funktorów logicznych realizujących zadaną funkcję [1], [18], [33]. Znajomość typu funkcji ma więc duże znaczenie praktyczne. Duże funkcje boolowskie o bardzo dużej liczbie argumentów (np. kilkaset) można poddać dekompozycji szeregowej lub równoległej, co sprawia, że poszczególne wyrażenia boolowskie są mniej skomplikowane, ale ciągle nieodpowiednie (ze względu na wielkość) w klasycznych procedurach wyznaczania widma. Stosując odpowiednio interpretowaną transformatę Walsha, można temu zaradzić.

Dyskretną transformację Walsha można stosować w sposób klasyczny, tak jak pokazano w rozdziale 5.8, lub w sposób szczególny – jeśli mamy do czynienia z funkcjami boolowskimi.

Binarna, m argumentowa funkcja boolowska f, opisana wektorem prawdy $\boldsymbol{y}_f$, może być traktowana jako specyficzna funkcja skalarna, jeśli jest zdefiniowana jako jednowyjściowa:

$$f : \{0,1\}^m \to \{0,1\}^1 \tag{6.25}$$

lub jako funkcja wektorowa, jeśli funkcja f jest wielowyjściowa:

$$f : \{0,1\}^m \to \{0,1\}^k, \text{ gdzie } k \text{ jest liczbą wyjść.} \tag{6.26}$$

W ostatnim przypadku funkcję można także traktować jako skalarną, jeśli jej wartości przedstawione będą w postaci zagregowanej, tzn. zamiast opisu binarnego poszczególnych wyjść $y_0, y_1, ..., y_{k-1}$ funkcji zastosować opis w postaci równoważnika dziesiętnego:

$$y = \sum_{i=0}^{k-1} y_i 2^i, \quad y_i \in \{0,1\}, \tag{6.27}$$

$$x = \sum_{i=0}^{m-1} x_i 2^i, \quad x_i \in \{0,1\}, \tag{6.28}$$

gdzie y jest zagregowaną wartością zmiennych wyjściowych, a x – zagregowaną wartością binarnych zmiennych wejściowych $x_0, x_1, ..., x_{m-1}$ funkcji f:

$$x \in \boldsymbol{X} = \{0,1,...,2^m-1\} \quad , y \in \boldsymbol{Y} = \{0,1,...,2^k-1\}, \tag{6.29}$$

$$\bigwedge_{x \in \boldsymbol{X}} f : x \mapsto y = f(x) \in \boldsymbol{Y}. \tag{6.30}$$

Ma to znaczenie praktyczne, gdyż postać zagregowana (której trywialnym przypadkiem jest boolowska funkcja z jednym wyjściem) pozwala na jednolitą interpretację zarówno jedno-, jak i wielowyjściowych funkcji. Takie postępowanie daje możliwość ujednolicenia obliczeń widma Walsha. Widmo dowolnej funkcji dyskretnej f wyznaczyć można bezpośrednio z definicji, gdzie macierzą jądrową przekształcenia jest macierz Hadamarda $\boldsymbol{H}_m$. W literaturze spotkać można dwa sposoby wyznaczania widma Walsha funkcji boolowskiej [33].

W jednym przypadku wartości m argumentowej funkcji f, reprezentowanej wartościami wektora prawdy $\boldsymbol{y}_f = [y_0, y_1, ..., y_{2^m-1}]$, pozostają niezmienione. Otrzymujemy wtedy wektor współczynników widmowych oznaczany symbolem $\boldsymbol{r}_f = [r_0, r_1, ..., r_{2^m-1}]$.

Drugi sposób polega na zakodowaniu elementów y_i wektora prawdy funkcji f według formuły $y_i \leftarrow 1 - 2y_i$, czyli $\{0,1\} \rightarrow \{1,-1\}$. Otrzymujemy wtedy wektor współczynników widmowych oznaczany symbolem $\boldsymbol{s}_f = [s_0, s_1, ..., s_{2^m-1}]$. Między obydwoma typami widma zachodzi prosta wzajemnie jednoznaczna zależność: $s_0 = 2^m - 2r_0$, $s_{i\neq 0} = -2r_i$, $i = 1, ..., 2^m - 1$.

Widmo Walsha typu $\boldsymbol{r}$ boolowskiej funkcji wielowyjściowej można wyznaczyć na dwa sposoby, w zależności od tego, czy funkcję traktujemy jako wektorową, czy skalarną. Jeśli funkcja ma reprezentację wektorową, to wyznaczenie widma dokonuje się dla każdego wyjścia oddzielnie, a uzyskane wyniki są odpowiednio sumowane. W formie zagregowanej (funkcję traktujemy jako skalarną) widmo wyznaczyć można identycznie jak w przypadku funkcji jednowyjściowej. Obie metody wyznaczania widma prowadzą do identycznych wyników. Omówioną zasadę można zaprezentować za pomocą prostego przykładu. Wyniki obliczeń zebrano w tablicy 1.

Kolumny (1–6) tablicy 1 stanową opis (tabelę prawdy) pewnej funkcji boolowskiej $f : \{0,1\}^3 \rightarrow \{0,1\}^2$. Kolumna (7) zawiera wartości zagregowanego stanu wyjść. Kolumny (8) oraz (9) przedstawiają wartości współczynników widmowych, wyznaczone oddzielnie dla każdego z wyjść funkcji f_1 oraz f_0. Kolumna (10) zawiera wartości widma obliczone na podstawie zagregowanego stanu występującego w kolumnie (7). W kolumnie (11) pokazano inny sposób wyznaczania widma na podstawie znajomości widma obliczanego oddzielnie dla każdego z wyjść funkcji (z kolumn 8 i 9). Wyniki w kolumnie (11) są identyczne jak te podane w kolumnie (10). Współczynniki widmowe w kolumnie (10) można wyznaczyć dwojako: jednorazowo na podstawie wartości zagregowanych lub oddzielnie dla każdego wyjścia funkcji f, z końcowym sumowaniem poszczególnych wartości współczynników widmowych.

Tablica 1. Funkcja boolowska oraz sposób wyznaczania jej widma Walsha

1	2	3	4	5	6	7	8	9	10	11
$x = \sum_{i=0}^{2} x_i 2^i$	x_2	x_1	x_0	$f_1(x)$	$f_0(x)$	$f(x) = 2f_1(x) + f_0(x)$	$\boldsymbol{r}_{f_1(x)}$	$\boldsymbol{r}_{f_0(x)}$	$\boldsymbol{r}_{f(x)}$	$\boldsymbol{r}_{f(x)} = 2\boldsymbol{r}_{f_1(x)} + \boldsymbol{r}_{f_0(x)}$
0	0	0	0	0	1	1	4	4	12	$2 \times 4 + 4 = 12$
1	0	0	1	0	1	1	2	0	4	$2 \times 2 + 0 = 4$
2	0	1	0	1	0	2	0	0	0	$2 \times 0 + 0 = 0$
3	0	1	1	0	0	0	-2	0	-4	$2 \times (-2) + 0 = -4$
4	1	0	0	1	0	2	-2	0	-4	$2 \times (-2) + 0 = -4$
5	1	0	1	1	0	2	0	0	0	0
6	1	1	0	1	1	3	-2	4	0	$2 \times (-2) + 4 = 0$
7	1	1	1	0	1	1	0	0	0	0

Współczynniki widmowe funkcji boolowskiej mogą być wyznaczane inaczej. Jeśli elementy macierzy $\boldsymbol{H}_m$ oraz elementy wektora $\boldsymbol{y}_f$ przyjmują wartości z tego samego zbioru danych, np. ze zbioru $1, -1$ lub $0, 1$, to widmo można wyznaczyć następująco. Traktując wiersze (kolumny) macierzy $\boldsymbol{H}_m$ jako 2^m wymiarowe wektory $\boldsymbol{h}_i = [h_{i0}, h_{i1}, ..., h_{i2^m-1}]$, $h_{ij}, y_i \in \{0, 1\}$ lub $y_i \in \{1, -1\}$, $i = 0, 1, ...2^m - 1$, widmo można obliczyć jako różnicę ($\diamond_i$) zgodnych ($+$) oraz niezgodnych ($-$) wartości występujących na odpowiadających sobie pozycjach wektorów $\boldsymbol{h}_i$ oraz $\boldsymbol{y}_f$:

$$\diamond_i = K_i^+(\boldsymbol{h}_i, \boldsymbol{y}_f) - K_i^-(\boldsymbol{h}_i, \boldsymbol{y}_f), \tag{6.31}$$

gdzie:
$K_i^+(\boldsymbol{h}_i, \boldsymbol{y}_f) = \#\{i : h_{ij} = y_i, j = 0, 1, ..., 2^m - 1\}$ – liczba takich samych (zgodnych) wartości występujących na tych samych pozycjach wektorów $\boldsymbol{h}_i$ oraz $\boldsymbol{y}_f$.
$K_i^-(\boldsymbol{h}_i, \boldsymbol{y}_f) = \#\{i : h_{ij} \neq y_i, j = 0, 1, ..., 2^m - 1\}$ – liczba niezgodnych wartości występujących na tych samych pozycjach wektorów $\boldsymbol{h}_i$ oraz $\boldsymbol{y}_f$.

Niech macierz Walsha–Hadamarda z elementami $\{0, 1\}$ ma oznaczenie $\boldsymbol{H}_{01}$, a z elementami $\{1, -1\}$ – oznaczenie $\boldsymbol{H}_{11}$ [23]. Niech wektor prawdy pewnej funkcji boolowskiej ma postać $\boldsymbol{y}_{01} = [0, 1, 1, 0]^T$.

Ten sam wektor w wersji zakodowanej ma postać $\boldsymbol{y}_{11} = [1, -1, -1, 1]^T$. Stosując regułę (6.31), otrzymujemy:

$$\boldsymbol{H}_{01} \diamond \boldsymbol{y}_{01} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \diamond \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2^+ - 2^- = 0 \\ 2^+ - 2^- = 0 \\ 2^+ - 2^- = 0 \\ 4^+ - 0^- = 4 \end{bmatrix}, \tag{6.32}$$

$$\boldsymbol{H}_{11} \diamond \boldsymbol{y}_{11} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \diamond \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2^+ - 2^- = 0 \\ 2^+ - 2^- = 0 \\ 2^+ - 2^- = 0 \\ 4^+ - 0^- = 4 \end{bmatrix}. \tag{6.33}$$

Można zauważyć, że otrzymane w ten sposób współczynniki widmowe funkcji boolowskiej tworzą widmo Walsha–Hadamarda typu $\boldsymbol{r}$.

Widmo Walsha wybranych funkcji boolowskich ma specyficzne cechy diagnostyczne, co pozwala na wykrywanie nadmiarowości funkcji boolowskich lub odkrywanie charakteru funkcji, np. czy funkcja jest liniowa, afiniczna gięta, większościowa lub mniejszościowa. Wymienione cechy trudno odnaleźć, analizując bezpośrednio wektor prawdy szczególnie wtedy, gdy funkcja boolowska jest funkcją o dużej liczbie zmiennych wejściowych.

Lemat 6.1. *Niech wektor* $\boldsymbol{y}_f = [y_0, y_1, ..., y_{2^m-1}]$, $y_i \in \{0,1\}$ *będzie wektorem prawdy funkcji boolowskiej* f. *Niech widmo Walsha tej funkcji tworzy wektor* $\boldsymbol{r}_f = [r_0, r_1, ..., r_{2^m-1}]$. *Niech funkcja boolowska p opisana jest wektorem prawdy* $\boldsymbol{y}_p = [\boldsymbol{y}_f, \boldsymbol{y}_f]$, *wtedy jej widmo Walsha ma postać*:

$$\boldsymbol{r}_p = \boldsymbol{y}_p \cdot \boldsymbol{H}_m = [2\boldsymbol{r}_f, \boldsymbol{0}]. \tag{6.34}$$

Dowód:
Funkcja f ma widmo:

$$\boldsymbol{r}_f = \boldsymbol{y}_f \cdot \boldsymbol{H}_m. \tag{6.35}$$

Funkcja p ma więc następujące widmo:

$$\begin{aligned} &\boldsymbol{y}_p \cdot \boldsymbol{H}_{m+1} = [\boldsymbol{y}_f, \boldsymbol{y}_f] \cdot \begin{bmatrix} \boldsymbol{H}_m & \boldsymbol{H}_m \\ \boldsymbol{H}_m & -\boldsymbol{H}_m \end{bmatrix} = \\ &= \begin{bmatrix} \boldsymbol{y}_f \cdot \boldsymbol{H}_m + \boldsymbol{y}_f \cdot \boldsymbol{H}_m & \boldsymbol{y}_f \cdot \boldsymbol{H}_m - \boldsymbol{y}_f \cdot \boldsymbol{H}_m \end{bmatrix} = [2\boldsymbol{r}_f, \boldsymbol{0}]. \end{aligned} \tag{6.36}$$

Lemat 6.1 jest również słuszny dla widma Walsha typu $\boldsymbol{s}$. Lemat 6.1 rozstrzyga, jak wyznaczać widmo Walsha nawet bardzo dużych funkcji boolowskich, których wektor prawdy można przedstawić w postaci:

$$\boldsymbol{y}_p = \underbrace{[\boldsymbol{y}_f\ \boldsymbol{y}_f ... \boldsymbol{y}_f]}_{2^m - krotnie}. \tag{6.37}$$

Oznacza to, że jeżeli znane jest widmo $\boldsymbol{r}_f$ podstawowej funkcji f, to większa funkcja boolowska typu (6.37) ma widmo:

$$\boldsymbol{r}_p = [2^m \cdot \boldsymbol{r}_f, \boldsymbol{0}]\,. \tag{6.38}$$

Definicja 6.1. *Funkcję boolowską* $\boldsymbol{y}_k = f_k(\boldsymbol{x})$, $\boldsymbol{x} = [x_0, x_1, ..., x_{m-1}]$ *nazywamy liniową, jeżeli można ją przedstawić w postaci wielomianu pierwszego stopnia*:

$$f_k(x_0, x_1, ..., x_{m-1}) = a_0 x_0 \oplus a_1 x_1 \oplus a_2 x_1 \oplus ... \oplus a_{m-1} x_{m-1} \oplus c, \quad a_i, c \in \{0, 1\}, \tag{6.39}$$

gdzie $k = 2^m \cdot c + \sum_{i=0}^{m-1} a_i 2^i$.

Dla $c = 1$ funkcja f_k nazywana jest często funkcją afiniczną.

Można utworzyć $k = 2^{m+1}$ funkcji f_k, co jest spostrzeżeniem trywialnym. Z własności dyskretnych funkcji Walsha wynika ponadto, że wektor prawdy dowolnej funkcji f_k wyznaczyć można wprost z funkcji Walsha o numerze k:

$$\begin{aligned} \boldsymbol{y}_k &= \frac{1}{2}(\boldsymbol{1} - wal(k,t)), \qquad \text{dla } c = 0, \\ \boldsymbol{y}_k &= \frac{1}{2}(\boldsymbol{1} + wal(k,t)), \qquad \text{dla } c = 1, \end{aligned} \tag{6.40}$$

gdzie $t = 0, 1, ..., 2^m - 1$.

Twierdzenie 6.1. *Każda liniowa funkcja boolowska* $f_k(\boldsymbol{x})$, $\boldsymbol{x} = [x_0, ..., x_{m-1}]$, *której wektor prawdy kodowany jest według formuły* $\{0, 1\} \to \{1, -1\}$, *ma widmo Walsha–Hadamarda o następującym rozkładzie współczynników widmowych*:

$$s_i = \begin{cases} 2^m \cdot (-1)^c & dla \quad i = (k - c)/2 \\ 0 & w\ przeciwnym\ przypadku \end{cases}, \tag{6.41}$$

gdzie c *oraz* k *mają znaczenie jak w Definicji* 6.1 *oraz* $i = 0, 1, ..., 2^m - 1$.

Twierdzenie to można również wyrazić inaczej.

Twierdzenie 6.2. *Każda liniowa funkcja boolowska* $f_k(\boldsymbol{x})$, $\boldsymbol{x} = [x_0, ..., x_{m-1}]$, *której wektor prawdy nie jest kodowany, czyli* $\{0, 1\} \to \{0, 1\}$, *ma widmo Walsha–Hadamarda o następującym rozkładzie współczynników widmowych*:

$$r_i = \begin{cases} 2^{m-1} & dla \quad i = 0 \\ 2^{m-1} \cdot (-1)^{1-c} & dla \quad i = (k-c)/2 \\ 0 & w\ przeciwnym\ przypadku \end{cases}, \tag{6.42}$$

gdzie c oraz k mają znaczenie jak w Definicji 6.1.

Przykład 6.2. *Niech wektor prawdy arbitralnie wybranej funkcji boolowskiej* $f_6(x_0, x_1, x_2) = x_1\overline{x}_2 + \overline{x}_1 x_2 = x_1 \oplus x_2$ *ma postać* $y_6 = [0, 1, 1, 0, 0, 1, 1, 0]$. *Mamy* $m = 3$ *argumentową funkcję, a dla* $k = 6$ *równanie* $k = 2^m \cdot c + \sum_{i=0}^{m-1} a_i 2^i$ *jest prawdziwe tylko dla* $c = 0$. *Widma Walsha funkcji* f_6 *są więc następujące*:

$$\boldsymbol{s}_6 = [s_0, s_1, ..., s_7] = [0, 0, 0, 8, 0, 0, 0, 0], \tag{6.43}$$

$$\boldsymbol{r}_6 = [r_0, r_1, ..., r_7] = [4, 0, 0, -4, 0, 0, 0, 0]. \tag{6.44}$$

Zgodnie z Twierdzeniem 6.1 *lub Twierdzeniem* 6.2, *funkcja* f_6 *jest funkcją liniową.*

Twierdzenie 6.3. *Każda funkcja boolowska* $f(\boldsymbol{x})$, $\boldsymbol{x} = [x_0, x_1, ..., x_{m-1}]$, *której wektor prawdy kodowany jest według reguły* $\{0, 1, -\} \to \{1, -1, 0\}$ *i posiadająca jeden punkt nieokreśloności ("−"), może być rozszerzona do funkcji liniowej wtedy i tylko wtedy, gdy jej widmo Walsha–Hadamarda ma następujący rozkład*:

$$s_i = \begin{cases} (-1)^{1-c} & dla \quad i = 0 \\ (-1)^c \times (2^m - 1) & dla \quad i = (k-c)/2 \\ \pm 1 & w\ przeciwnym\ przypadku \end{cases}, \tag{6.45}$$

gdzie c oraz k mają znaczenie jak w Definicji 6.1 *oraz* $i = 0, 1, ..., 2^m - 1$.

Jeśli na podstawie Twierdzenia 6.3 funkcja f może być rozszerzona do funkcji liniowej, to punkt, w którym jest ona nieokreślona, powinien zostać zastąpiony wartością 0, dla $c = 1$, lub wartością 1, dla $c = 0$.

Przykład 6.3. *Sprawdzić, czy funkcje boolowskie* f *oraz* g, *reprezentowane wektorami prawdy* $\boldsymbol{y}_f = [0, 1, 1, 0, -, 1, 1, 0]$ *oraz* $\boldsymbol{g}_f = [1, 1, 1, 0, 1, 1, 0, -]$, *mogą być rozszerzone do funkcji liniowej.*

Po zakodowaniu opisanych funkcji ich wektory prawdy mają następującą postać $\boldsymbol{y}_f^{cod} = [1,-1,-1,1,0,-1,-1,1]$ *oraz* $\boldsymbol{g}_f^{cod} = [-1,-1,-1,1,-1,-1,1,0]$. *Współczynniki widmowe Walsha–Hadamarda zakodowanych funkcji* f *i* g *tworzą wektory*: $\boldsymbol{s}_f = [-1,-1,-1,7,1,1,1,1]$ *oraz* $\boldsymbol{s}_g = [-3,-1,-5,1,-1,-3,1,3]$. *Zgodnie z Twierdzeniem* 6.3, *można przyjąć, że współczynniki widmowe powinny przyjmować jedną z następujących wartości:* $s_0 \pm 1$, $s_i = \pm 7$. *Pozostałe współczynniki powinny przyjmować wartości* ± 1. *Wynika z tego, że funkcja* f *może, a funkcja* g *nie może być rozszerzona do funkcji liniowej.*

Propozycja 6.1. *Niech* $T_f = \{\boldsymbol{x} \in \{0,1\}^m : f(\boldsymbol{x}) = 1\}$ *oraz* $F_f = \{\boldsymbol{x} \in \{0,1\}^m : f(\boldsymbol{x}) = 0\}$. *Niech* f *będzie częściowo określona, czyli niezdefiniowana w* d *punktach. Funkcja boolowska* f *reprezentowana wektorem prawdy* $\boldsymbol{y}_f = [y_0, y_1, ..., y_{2^{m-1}}]$, $y_i \in \{+1,-1,0\}$ *może być rozszerzona do funkcji liniowej*:

1. *Jeśli* $s_0 = +d \to \{-\} \in T_f$, *co oznacza, że punkty nieokreślone powinny zostać zatąpione wartościami* 1.
2. *Jeśli* $s_0 = -d \to \{-\} \in F_f$, *co oznacza, że punkty nieokreślone powinny zostać zastąpione wartościami* 0.
3. *Jeśli* $s_0 \neq d \to \{-\} \in T_f \cup F_f$, *to dla pewnego* i, $s_i = \pm(2^m - d) = (-1)^c \cdot (2^m - d)$.

Twierdzenie 6.4. *Każda* m *argumentowa boolowska funkcja gięta ma następujący rozkład współczynników widmowych*:

$$s_i = \pm 2^{m/2}, \qquad i = 0, 1, ..., 2^m - 1. \tag{6.46}$$

Łatwo zauważyć, że funkcje gięte można konstruować tylko dla parzystych m. Znając m argumentową funkcję giętą, można znaleźć funkcję boolowską dla $m+2$ argumentów jako złożenie dwóch mniejszych boolowskich funkcji giętych.

Propozycja 6.2. *Niech zbiór* B_{m-2} (*dla parzystego* m) *będzie zbiorem funkcji giętych* $f : \{0,1\}^{m-2} \to \{0,1\}$. *Niech* f_a, $f_b \in B_{m-2}$, *wtedy funkcja* f_c:

$$f_c(x_0, x_1, ..., x_{m-1}) = \begin{cases} f_a(x_0, x_1, ..., x_{m-3}), & x_{m-2} = 0, \quad x_{m-1} = 0 \\ f_a(x_0, x_1, ..., x_{m-3}), & x_{m-2} = 0, \quad x_{m-1} = 1 \\ f_b(x_0, x_1, ..., x_{m-3}), & x_{m-2} = 1, \quad x_{m-1} = 0 \\ f_b(x_0, x_1, ..., x_{m-3}) \oplus 1, & x_{m-2} = 1, \quad x_{m-1} = 1 \end{cases} \tag{6.47}$$

jest funkcją giętą.

Przykład 6.4. *Niech wektory* $\boldsymbol{y}_a = [0,1,1,1]$ *oraz* $\boldsymbol{y}_b = [0,1,0,0]$ *są wektorami prawdy funkcji boolowskich* f_a *i* f_b*. Zgodnie z Propozycją* 6.2*, można skonstruować wektor prawdy* $\boldsymbol{y}_c = [0,0,0,1,1,1,1,0,1,1,0,1,1,1,0,1]$ *funkcji* f_c*. Widmo Walsha–Hadamarda funkcji* f_c *jest następujące:*
$\boldsymbol{s}_c = [-4,4,-4,-4,4,4,4,-4,4,-4,4,4,4,4,4,-4]$.
Z Twierdzenia 6.4 *wynika, że funkcja* f_c *jest funkcją giętą.*

Przedstawiony sposób wyznaczania widma funkcji boolowskiej jest niepraktyczny. Dla dużych funkcji boolowskich nawet szybka transformata Walsha jest nieprzydatna ze względu na złożoność problemu. Dla takich funkcji należy znaleźć inną ich reprezentację, umożliwiającą przeprowadzenie obliczeń.

Z elementów wektora prawdy można utworzyć dwa następujące podzbiory:

$$\begin{aligned} ON &= \{\boldsymbol{x} \in \{0,1\}^m : f(\boldsymbol{x}) = 1\}, \\ DC &= \{\boldsymbol{x} \in \{0,1\}^m : f(\boldsymbol{x}) = \text{"}-\text{"}\}. \end{aligned} \tag{6.48}$$

gdzie znak "$-$" to oznaczenie wartości nieistotnej (jeszcze nieokreślonej). Pozostałe elementy wektora prawdy przyjmują wartości 0.

Elementy te tworzą zbiór $OFF = \{\boldsymbol{x} \in \{0,1\}^m : f(\boldsymbol{x}) = 0\}$. Zbiór ten nie będzie wykorzystywany, gdyż pozostałe dwa jednoznacznie definiują funkcję f. Elementy podzbiorów ON oraz DC można łączyć w tzw. kostki (ang. *cubes*). Kostki są rozłączne, czyli się nie przecinają (ang. *disjoint*), jeśli nie mają żadnego wspólnego mintermu [18, 19, 29]. Odpowiednie kostki będą oznaczane identycznie jak zbiory (6.48). Elementy wektora prawdy funkcji boolowskiej f pozostają w pierwotnej postaci, czyli nie są kodowane.

Niech k jest liczbą rozłącznych kostek ON. Niech w jest liczbą rozłącznych kostek DC. Niech p jest liczbą elementów "$-$" w danej kostce. Liczba argumentów funkcji f będzie oznaczana literą m. Numer kostki, dla której wyznaczane jest widmo Walsha, oznaczane będzie wskaźnikiem j. Współczynniki widmowe $\boldsymbol{s}_f = [s_0, s_1, ..., s_{2^m-1}]$ funkcji f wyznacza się oddzielnie dla kostek ON oraz DC.

$$s_{0_j}^{ON} = 2^m - 2 \cdot 2^p, \tag{6.49}$$

$$s_{0_j}^{DC} = 2^{m-1} - 2^p. \tag{6.50}$$

Następnie porządkujemy sukcesywnie zmienne wejściowe funkcji $f(x_1, x_2, ..., x_m)$ w porządku leksykograficznym i dla każdej zmiennej w kostce j podstawiamy odpowiednią wartość zmiennej w tej kostce. Jeśli wartość odpowiedniej zmiennej jest nieokreślona, wtedy wartość wszystkich współczynników widmowych, gdzie występuje taka zmienna, wynosi 0. Widmowe współczynniki cząstkowe wylicza się według następującego schematu:

$$s_{1j}^{ON} = \begin{cases} (-1)^{1+x_1} \cdot (2 \cdot 2^p) & \text{dla } x_1 \neq \text{"}-\text{"} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{6.51}$$

$$s_{1j}^{DC} = \begin{cases} (-1)^{1+x_1} \cdot 2^p & \text{dla } x_1 \neq \text{"}-\text{"} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{6.52}$$

$$s_{2j}^{ON} = \begin{cases} (-1)^{1+x_2} \cdot (2 \cdot 2^p) & \text{dla } x_2 \neq \text{"}-\text{"} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{6.53}$$

$$s_{2j}^{DC} = \begin{cases} (-1)^{1+x_2} \cdot 2^p & \text{dla } x_2 \neq \text{"}-\text{"} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{6.54}$$

$$s_{12j}^{ON} = \begin{cases} (-1)^{(1+x_1+x_2)} \cdot (2 \cdot 2^p) & \text{jeśli } x_1 \neq \text{"}-\text{"} \ \wedge \ x_2 \neq \text{"}-\text{"} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{6.55}$$

$$s_{12j}^{DC} = \begin{cases} (-1)^{(1+x_1+x_2)} \cdot 2^p & \text{jeśli } x_1 \neq \text{"}-\text{"} \ \wedge \ x_2 \neq \text{"}-\text{"} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{6.56}$$

...

$$s_{123...nj}^{ON} = \begin{cases} (-1)^{1+\sum_{m=1}^{n} x_m} \cdot (2 \cdot 2^p) & \text{dla } x_1, x_2, ..., x_n \neq \text{"}-\text{"} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{6.57}$$

$$s_{123...nj}^{DC} = \begin{cases} (-1)^{1+\sum_{m=1}^{n} x_m} \cdot 2^p & \text{dla } x_1, x_2, ..., x_n \neq \text{"}-\text{"} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \tag{6.58}$$

Poszczególne elementy widma Walsha $\boldsymbol{s} = [s_0, s_1, ..., s_{2^m-1}]$ wyznacza się na podstawie następujących formuł:
zerowy współczynnik:

$$s_0 = \left(\sum_{j=1}^{w+k} s_{0_j} \right) - (k-1) \times 2^m - w \times 2^{m-1}, \tag{6.59}$$

a pozostałe współczynniki:

$$s_i = \left(\sum_{j=1}^{w+k} s_{i_j} \right). \tag{6.60}$$

Opisane powyżej zasady można przedstawić w postaci programu komputerowego.

Przykład 6.5. *Należy wyznaczyć współczynniki widmowe Walsha funkcji boolowskiej f opisanej wektorem prawdy $\boldsymbol{y}_f = [0, 1, -, 1, 1, -, 1, -]$. Kodując elementy wektora $\boldsymbol{y}_f$ zgodnie ze schematem $\{0, 1, -\} \rightarrow \{1, -1, 0\}$, można wyznaczyć widmo Walsha klasycznie, wykorzystując do tego celu macierz jądrową Hadamarda $\boldsymbol{H}_3$. Można jednak zastosować zbiór reguł (6.49)–(6.60), których efekty działania zaprezentowano w tablicy 2. Zamiast wektora prawdy w obliczeniach zastosowano kostki rozłączne ON oraz DC.*

Tablica 2. Widmo Walsha funkcji boolowskiej reprezentowanej przez rozłączne kostki

x_1	x_2	x_3	Typ kostki	s_{0_j}	s_{1_j}	s_{2_j}	s_{12_j}	s_{3_j}	s_{13_j}	s_{23_j}	s_{123_j}	Numer kostki
0	–	1	*ON*	4	−4	0	0	4	4	0	0	$j = 1$
1	–	0	*ON*	4	4	0	0	−4	4	0	0	$j = 2$
0	1	0	*DC*	3	−1	1	1	−1	−1	1	1	...
1	–	1	*DC*	2	2	0	0	2	−2	0	0	$j = w + k$
s				**−3**	**1**	**1**	**1**	**1**	**5**	**1**	**1**	

Podany w tablicy 2 schemat obliczeń współczynników widmowych sprawia, że ich wyznaczenie jest mniej czasochłonne, szczególnie dla dużych funkcji boolowskich, gdzie bezpośrednie zastosowanie macierzy Hadamarda, ze względu na jej duże rozmiary, nie jest możliwe.

Współczynniki widmowe w tablicy 2 są uporządkowane w tzw. porządku Paleya, różnym od porządku Walsha–Hadamarda.

Porządek widma można jednak zmieniać przez stosowanie macierzy permutujących [24]. Dla dużych funkcji boolowskich (kilkadziesiąt zmiennych wejściowych) kostki rozłączne generuje się wyłącznie maszynowo, za pomocą specjalizowanych programów, np. ESPRESSO [18, 19]. Taka reprezentacja, zapisana dodatkowo w specjalnym formacie, sprawia, że opis funkcji staje się zwarty.

Program 6.3. *Program w Matlabie, za pomocą którego można obliczyć komplet współczynników widmowych funkcji boolowskiej reprezentowanej rozłącznym zbiorem kostek ON i DC. Program jest szczególnie przydatny dla dużych funkcji boolowskich, gdzie wykorzystanie macierzy Hadamarda ze względu na ich duży rozmiar nie jest możliwe. Prezentowane rozwiązanie zostało zaimplementowane w postaci funkcji Matlaba.*

```
function f=GetCoeff(x)    % Definicja funkcji
if exist(x,'file')==0
   disp('Podaj prawidlowy plik z danymi');
   return
end
id=fopen(x);          % Funkcja otwierajaca plik z opisem funkcji
[pathstr,name,ext]=fileparts(x);
```

```
fidd=fopen('temp.pla','w'); e
while ~feof(fid);
      tline=fgets(fid);
      if strcmpi(tline(1), '.')
      else
         fwrite(fidd,tline);
      end
end
fclose all;
delete(x);
if ismac
   movefile('temp.pla',strcat(pathstr,'/',name,ext));
else
   movefile('temp.pla',strcat(pathstr,'\',name,ext));
end
function g=encode(char)   % Kodowanie {0,1,-} -> {1,-1,0}
if char=='0'
   g=1;
elseif char=='1'
   g=-1;
elseif char=='-'
   g=0;
end
end
fileid=fopen(x); %Otwarcie pliku z opisem funkcji
d=textscan(fileid, '%s %s');
fclose(fileid); % Zamkniecie pliku
m=length(d{1}{1});
ss=length(d{2}{1});
fileid=fopen(x);
dd=textscan(fileid, strcat('%s ', repmat('%c',1,ss)));
```

```
fclose(fileid);
r=0;
h=0;
for i=1:length(d{1})
     h=numel(find(ismember(dd{1}{i},'-')));
     if h == 1
        r=r+1;
     end
end
war = 2.^ m - r;
aff=0;
for q=2:ss+1
    k=numel(find(ismember(dd{q},'1')));   % Liczba kostek ON
    w=numel(find(ismember(dd{q},'~')));   % Liczba kostek DC
    temp_matrix=zeros(w+k+1,(2^m));
    for j=1:w+k
        p=sum(d{1}{j}=='-');
        for i=1:2^m
            if i == 1
               if strcmpi(dd{q}(j), '1')
                  temp_matrix(j,1)=2^m - 2*(2^p);
               elseif strcmpi(dd{q}(j),'~')
                  temp_matrix(j,1)=2^(m-1) - 2^p;
               end
            else
               if strcmpi(dd{q}(j),'1')
                  seq=0;
                  for x=1:m
                      if and(strcmpi(d{1}{j}(x),'-'),
                         (bitand((i-1),2^(x-1))>0))
                          temp_matrix(j,i)=0;
```

```
                    break
                end
                if bitand((i-1), 2^(x-1))>0
                    seq=seq + xor(1,d{1}{j}(x));
                    seq=seq + d{1}{j}(x);
                end
                temp_matrix(j,i)=((-1)^(1+seq))*(2*2^p);
            end
        elseif strcmpi(dd{q}(j),'~')
            seq=0;
            for x=1:m
                if and(strcmpi(d{1}{j}(x),'-'),
                  (bitand((i-1), 2^(x-1))>0))
                  temp_matrix(j,i)=0;
                  break
                end
                if bitand((i-1),2^(x-1))>0
                   seq=seq+d{1}{j}(x);
                end
                temp_matrix(j,i)=(-1)^(1+seq)*(2^p);
            end
        end
      end
   end
end
for j=1:(2^m)
    if j==1
       temp_matrix(w+k+1,j)=
       sum(temp_matrix(1:w+k,j))-(k-1)*(2^m)-(w*(2^(m-1)));
    else
       temp_matrix(w+k+1,j)=sum(temp_matrix(1:w+k,j));
```

```
          end
      end
      temp_matrix(w+k+1,1:2^m);
      vec=ismember(temp_matrix(w+k+1,1:2^m),[war,-war]);
      if numel(find(ismember( vec, 1)))>0
         aff=aff+1;
      end
      spec='';
      for i=1:(2^m)
          str=sprintf('[%d]',temp_matrix(w+k+1,i));
          spec=strcat(spec, str);
      end
      disp('spectrum:');
      disp(spec);
  end
end
```

Powyższy program pozwala na wyznaczenie widma funkcji boolowskiej, która opisana jest za pomocą rozłącznych kostek ON oraz DC. Znajomość widma pozwala z kolei na klasyfikację funkcji, np.: liniowe, afiniczne lub gięte, oraz rozstrzyga, w przypadku funkcji słabo określonej, czy można tak uzupełnić stany nieokreślone, aby rozszerzyć funkcję do postacji liniowej.

6.4. Porządkowanie Binarnego Diagramu Decyzyjnego

Binarne Diagramy Decyzyjne (ang. *Binary Decision Diagrams*) – BDD są jedną z ważnych form reprezentacji wyrażeń boolowskich w pamięci komputera i efektem redukcji drzew decyzyjnych funkcji [6], [33]. Stosuje się je jako narzędzie syntezy logicznej. Binarny Diagram Decyzyjny jest w zasadzie strukturą danych reprezentującą funkcję dyskretną. W przypadku małych funkcji można go przedstawiać również graficznie. Reprezentacja funkcji boolowskiej w postaci BDD pozwala na efektywne wykonywanie działań na wyrażeniach

boolowskich i doczekała się licznych implementacji algorytmicznych, których szczegółowe opisy można znaleźć w pracach [2, 33]. Dodatkowym atutem reprezentacji w postaci BDD jest możliwość zapisywania za ich pomocą nie tylko funkcji binarnych, ale także funkcji wielowartościowych i wielowyjściowych.

Istnieje wiele postaci diagramów decyzyjnych, w zależności od sposobu dekomponowania podstawowej funkcji f. W niniejszej książce przedstawione zostaną podstawowe formuły dekompozycji, ale, co należy podkreślić, jej celem nie jest analiza właściwości BDD. Celem wywodu jest pokazanie, jak konstruować optymalnie uporządkowany BDD, korzystając z własności widma Walsha. Wypadkowy rozmiar BDD uzależniony jest od uporządkowania występujących w nim zmiennych funkcji boolowskiej i zależy od właściwości funkcji (np. funkcje symetryczne, nie w pełni określone, wielopoziomowe funkcje AND/OR/EXOR itp.). Znalezienie optymalnego uporządkowania zmiennych jest problemem NP-trudnym, gdyż złożoność czasowa poszukiwania optymalnego BDD wynosiłaby $O(n!2^n)$, co już dla niewielkich n jest liczbą bardzo dużą.

Definicja 6.2. *Binarny diagram decyzyjny reprezentujący funkcję boolowską f jest acyklicznym grafem $G = (V, E)$, ze zbiorem węzłów V oraz zbiorem krawędzi E. Graf składa się z terminalnych węzłów o wartościach 0 i 1 oraz etykietowanych węzłów nieterminalnych. Funkcja f tworzy diagram na podstawie wybranych reguł dekompozycji funkcji f.*

Twierdzenie 6.5. *Każdą funkcję boolowską $f(x_0, ..., x_{n-1})$ można zdekomponować, stosując rekurencyjnie jeden z następujących schematów:*

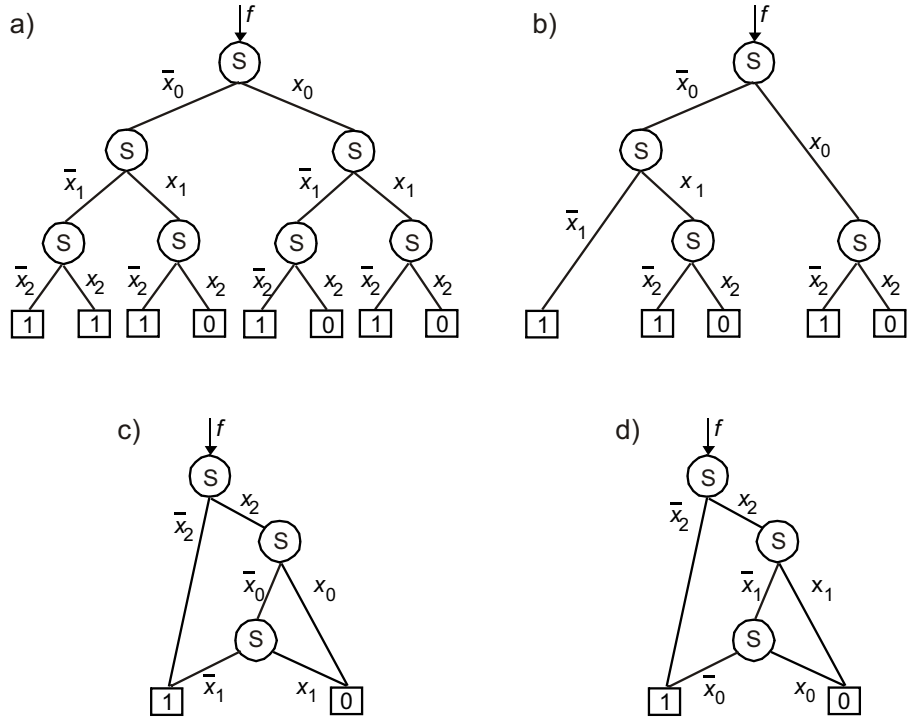
$$\begin{aligned} pD &: f(x_0, ..., x_{n-1}) = f_0 \oplus x_0 f_2 \\ nD &: f(x_0, ..., x_{n-1}) = \overline{x}_0 f_2 \oplus f_1 \\ S &: f(x_0, ..., x_{n-1}) = \overline{x}_0 f_0 \oplus x_0 f_1 \end{aligned} \quad , \tag{6.61}$$

gdzie: $f_0 = f(0, x_1, ..., x_{n-1})$, $f_1 = f(1, x_1, ..., x_{n-1})$, $f_2 = f_0 \oplus f_1$.

Podane w Twierdzeniu 6.5 schematy nazywane są dekompozycją *Positive Davio* (pD), *Negative Davio* (nD) oraz dekompozycją *Shannona* (S). Za optymalny uznajemy taki BDD, który ma najmniejszą liczbą węzłów.

Przykład 6.6. *Dana jest funkcja boolowska* $f(x_0, x_1, x_2) = \overline{x}_0\overline{x}_1 + \overline{x}_2$. *Binarne drzewo Shanonna funkcji* f *prezentuje rys.* 31a.

Pozostałe diagramy na rysunku przedstawiają różne typy BDD funkcji f *dla różnych porządków zmiennych wejściowych. BDD z najmniejszą liczbą węzłów uzyskano dla porządku zmiennych* (x_2, x_0, x_1) *lub* (x_2, x_1, x_0), *co prezentują diagramy na rys.* 31c *oraz* 31d.



Rys. 31. Binarne drzewo decyzyjne (a) BDD dla różnych uporządkowań zmiennych (b–d)

Dla dużych funkcji boolowskich znalezienie minimalnego BDD nie jest proste. W celu rozwiązania zadania stosowane są heurystyki, za pomocą których wskazać można przynajmniej suboptymalne struktury Diagramów Decyzyjnych.

Jedną z metod znajdowania suboptymalnych BDD funkcji boolowskiej jest analiza widma Walsha uzyskanego za pomocą rozłącznych kostek, o których była już mowa w poprzednim podrozdziale.

Można zauważyć, że funkcję boolowską f można opisać współczynnikami:

$$f(x_1, ..., x_n) = \frac{1}{2^{n+1}} \left[2^n - s_0 - \sum_{a=1}^{2^n-1} s_a (-1)^{x_1^{a_1} \oplus x_2^{a_2} \oplus ... \oplus x_n^{a_n}} \right], \tag{6.62}$$

gdzie: $a_1, a_2, ..., a_n \in \{0, 1\}$ są bitami binarnej reprezentacji liczby a oraz $x^{a_i=0} = 0$ i $x^{a_i=1} = 1$.

Analizując wzór (6.62), można spostrzec, że współczynnikom widmowym $s_1, ..., s_{2^n-1}$ przypisano jawną korelację ze zmiennymi wejściowymi funkcji boolowskiej f:

$$s_1 \leftrightarrow x_1, \quad s_{12} \leftrightarrow x_1 \oplus x_2, \quad s_{123} \leftrightarrow x_1 \oplus x_2 \oplus x_3, ..., s_{12...n} \leftrightarrow x_1 \oplus x_2 \oplus ... \oplus x_n. \tag{6.63}$$

Ten sposób zapisu oznacza, że można grupować współczynniki widmowe. Współczynniki oznaczone jedną cyfrą są współczynnikami pierwszego rzędu, oznaczone dwoma cyframi to współczynniki drugiego rzędu itd. Okazuje się, że znajomość współczynników pierwszego rzędu może być miarą uporządkowania Binarnego Diagramu Decyzyjnego [26]. Ponieważ struktura Diagramów Decyzyjnych jest przeważnie formowana dla dużych funkcji boolowskich, poniżej przedstawiono algorytm generowania współczynników widmowych pierwszego rzędu, gdzie rozpatrywany jest opis funkcji za pomocą rozłącznych kostek ON oraz DC. Realizacja takiego zadania wymaga wcześniejszego przygotowania pliku tekstowego, który zawierać będzie opis funkcji boolowskiej. Pliki tego typu mają zwyczajowo rozszerzenie pla lub txt, ale z praktycznego punktu widzenia jest to bez znaczenia. Zawartość pliku, jako dane wejściowe, jest następnie analizowana w przebiegu algorytmu. Plik opisujący przykładową funkcję boolowską za pomocą rozłącznych kostek może wyglądać następująco:

$$\begin{array}{cccc} 0 & 0 & 1 & ON \\ - & - & 0 & ON \end{array}$$

Jest to ta sama funkcja f, którą przywołano w przykładzie 6.6. Jest to funkcja w pełni określona, zatem wystarczy podać informację, dla jakich wartości zmiennych wejściowych funkcja przyjmuje wartość 1. Dla pozostałych kombinacji funkcja przyjmuje wartość 0. Poniżej przedstawiono program generowania współczynników pierwszego rzędu na podstawie pliku wsadowego zawierającego opis funkcji w postaci rozłącznych kostek.

Program 6.4. *Funkcja w Matlabie, za pomocą której można wyznaczyć współczynniki widmowe Walsha pierwszego rzędu, na podstawie opisu funkcji w postaci rozłącznych kostek ON i DC.*

```
function f=Getcoeff(x)  % Definicja funkcji
if exist(x,file')==0    % Sprawdzenie poprawnosci pliku wsadowego
   disp('podaj prawidlowy plik z danymi');
   return
end
fileid = fopen(x);              % Otwarcie pliku z opisem funkcji
d = textscan(fileid, '%s %s');
fclose(fileid);                    % Zamkniecie pliku.
size_of_cube = length(d{1}{1});    % Liczba elementow kostki
number_of_cubes = length(d{1});    % Liczba rozlacznych kostek
for i=1:size_of_cube
    s(i)=0;
end
for j=1:number_of_cubes % Liczba elementow w kostce typu DC
     p=0;
     for i=1:size_of_cube
         if strcmp(d{1}{j}(i),'-')
            p=p+1;
     end
end
for i=1:size_of_cube        % Wyznaczenie wsp. widmowych I rzedu
    if strcmpi(d{2}{j},'ON')
```

```
        v=power(2,(p+1));
        if d{1}{j}(i)=='0'
            s(i)=s(i)-v;
        end
        if d{1}{j}(i)=='1'
            s(i)=s(i)+v;
        end
    end
    if strcmpi(d{2}{j},'DC')
        v=2^(p);
        if d{1}{j}(i)=='0'
            s(i)=s(i)-v;;
        end
        if d{1}{j}(i)=='1'
            s(i)=s(i)+v;
        end
    end
end
end f=fliplr(s);
```

Współczynniki widmowe pierwszego rzędu można porządkować według reguł zaprezentowanych w tablicy 3.

Tablica 3. Sposoby porządkowania widma Walsha pierwszego rzędu

Opis porządku	Symbol	Przykład
Narastająco	$>$	$s_1 > ... > s_{n-1} > s_n$
Malejąco	$<$	$s_1 < ... < s_{n-1} < s_n$
Narastająco bezwzględnie	$\|>\|$	$s_1 \|>\| ... \|>\| s_{n-1} \|>\| s_n$
Malejąco bezwzględnie	$\|<\|$	$s_1 \|<\| ... \|<\| s_{n-1} \|<\| s_n$

Porządek zmiennych BDD ustala się na podstawie wartości współczynników widmowych. Im większa jego wartość, tym odpowiadająca mu zmienna wejściowa jest ważniejsza i od niej rozpoczyna się budowę diagramu. Jeśli wartości współczynników są równe, wybór zmiennej inicjującej jest arbitralny.

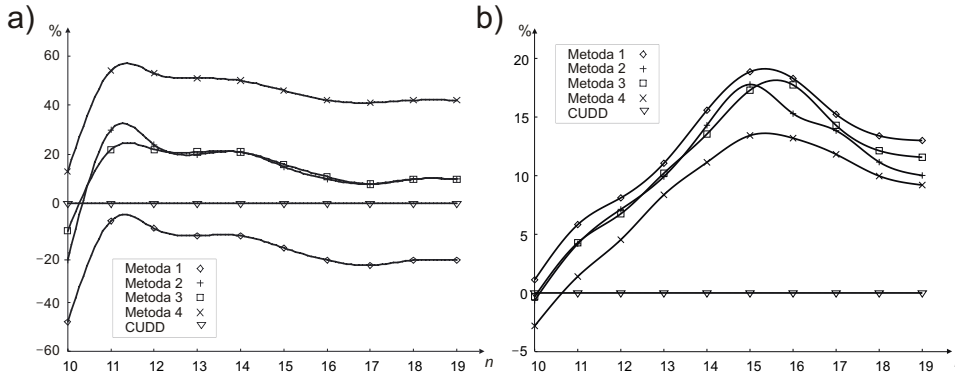
W praktyce należy rozważyć następujące metody porządkowania BDD:

- metoda 1: $min\{>, <, |>|, |<|\}$,
- metoda 2: $min\{>, <\}$,
- metoda 3: $min\{|>|, |<|\}$,
- metoda 4: $random\{>, <, |>|, |<|\}$,

gdzie *min* oznacza minimalną liczbę węzłów BDD uzyskaną w zbiorze różnych porządków zmiennych wejściowych, a *random* jest wyborem losowym. Na potrzeby eksperymentów wygenerowano losowo 300 funkcji boolowskich dla $n = 10, 11, ..., 20$ zmiennych wejściowych, które poddano przekształceniom do postaci BDD. Liczbę zmiennych funkcji boolowskiej ograniczono ze względu na możliwości pakietu CUDD, który służy do generowania suboptymalnych BDD [26, 32].

Wyrażony w procentach zysk czasowy porządkowania BDD oraz procentowa redukcja liczby węzłów BDD w stosunku do wyników uzyskiwanych w pakiecie CUDD zostały przedstawione na rys. 32. Ponieważ w praktyce występują funkcje boolowskie o większej liczbie zmiennych wejściowych, zbadano również takie przypadki, co obrazuje tablica 4. Zastosowano następujące oznaczenia: #i – liczba wejść układu, #o – liczba wyjść układu. Pozostałe kolumny zawierają informacje na temat czasów porządkowania BDD oraz liczby utworzonych różnymi metodami węzłów suboptymalnych BDD.

Ostatnia kolumna tablicy zawiera informację na temat najlepszego (z minimalną liczbą węzłów) BDD otrzymanego metodą widmową wraz z sumarycznym czasem poszukiwań różnych wersji BDD. Analiza wyników wskazuje, że metoda widmowa pozwala na tworzenie bardziej oszczędnych struktur BDD w stosunku do struktur uzyskiwanych za pomocą pakietu CUDD.



Rys. 32. Wyrażony w procentach czas wyznaczania struktury BDD (a) oraz procentowa mniejsza liczba węzłów BDD dla różnych sposobów porządkowania zmiennych wejściowych w stosunku do wyników pakietu CUDD (b)

Tablica 4. Efekty porządkowania zmiennych za pomocą pakietu CUDD oraz metodą widmową (liczba węzłów/czas w sek.)

Układ	#i	#o	CUDD/s	$>/s$	$\|>\|/s$	$</s$	$\|<\|/s$	$min/tot(s)$
9sym	9	1	25/.02	25/.02	25/.02	25/.02	25/.02	25/.06
co14	14	1	27/.02	27/.02	27/.02	27/.02	27/.02	27/.04
dist	8	5	121/.02	121/.02	121/.02	121/.02	121/.02	121/.08
ex1010	10	10	1060/.14	1058/.08	1054/.08	1056/.08	1050.08	1050.32
ex5	8	63	242/.05	186/.02	205/.03	204/.03	181/.02	181/.01
majority	5	1	8/.02	8/.02	8/.02	8/.02	9/.00	8/.06
mlp4	8	8	135/.05	145/.05	151/.05	151/.05	145/.05	145/.202
sqar5	5	8	35/.02	33/.02	34/.02	33/.02	33/.02	33/.06
sym10	10	1	31/.03	31/.02	31/.02	31/.02	31/.02	31/.08
Total			**1684/.37**	**1634/.27**	**1656/.26**	**1656/.26**	**1622/.21**	**1621/.91**

7. Zakończenie

W książce omówione zostały zagadnienia związane z przetwarzaniem sygnałów cyfrowych, a niezbędne wywody matematyczne objaśnione zostały licznymi przykładami numerycznymi i programami w języku Matlab.

Otaczającą nas rzeczywistość próbuje się opisywać za pomocą różnych modeli matematycznych, które w możliwie najprostszy sposób powinny tę rzeczywistość przybliżać. Posługiwanie się modelami matematycznymi ma wiele zalet, gdyż w zależności od potrzeb te same zjawiska fizyczne można opisywać różnymi modelami. Modele matematyczne pozwalają z kolei na zrozumienie zjawisk występujących w przyrodzie, gdzie mamy do czynienia ze zjawiskami o charakterze ciągłym. Sygnał cyfrowy jest tylko przybliżeniem sygnału analogowego. Ten substytut, chociaż niedoskonały, pozwala na wystarczająco precyzyjną budowę algorytmów przetwarzania sygnałów, które usuwają zniekształcenia, zakłócenia i niedokładności sygnału rejestrowanego przez urządzenia rejestrujące.

Istnieje wiele metod wydobywania informacji z przetwarzanego sygnału. Bez wątpienia należą do nich metody transformowania sygnału oryginalnego, przekształconego do postaci cyfrowej, do różnych przestrzeni. Otrzymane w ten sposób widmo sygnału, reprezentujące rozkład energii w dziedzinie czasowo-częstotliwościowej, nie zawsze ukazuje obserwatorowi wystarczająco przejrzyście cechy sygnału i wymaga niekiedy trudnej interpretacji zjawisk występujących w sygnale. Tym samym część, niekiedy ważnych z poznawczego punktu widzenia, cech sygnału zostaje utracona. Łącząc algorytmy przetwarzania sygnałów cyfrowych z metodami przetwarzania obrazów, uzyskujemy możliwość odkrywania cech, które są w obrazie ukryte. Cechy te możemy wtedy łatwiej analizować, a nawet obrazować graficznie. Wymienione zagadnienia zostały w niniejszej monografii opisane i wyjaśnione za pomocą wielu kompletnych przykładów numerycznych.

Literatura

[1] Afshord S. T., Pottosin Y., Arasteh B., *An input variable partitioning algorithm for functional decomposition of a system of boolean functions based on the tabular method*, Discrete Applied Mathematics, 185, 208–219, 2015.

[2] Astola J., Stanković R., Fundamentals of Switching Theory and Logic Design. A Hands on Approach, Springer-Verlag, Dordrecht, the Netherlands, 2006.

[3] Basztura C., Źródła, sygnały i obrazy akustyczne, WKŁ, Warszawa, 1988.

[4] Beauchamp K., Walsh Functions and their applications, Academic Press, New York, 1975.

[5] Białasiewicz J., Falki i aproksymacje, WNT, Warszawa, 2000.

[6] Bibilo P. N., Leonczyk P. V., *Decomposition of systems of boolean functions determined by binary decision diagrams*, Journal of Computer and Systems Sciences International, 50, 609–624, 2011.

[7] Biedrońska M., Grzymkowski R., Korek K., Nawrat A., Funkcje gięte, R-funkcje Rwaczewa, funkcje Walsha i ich zastosowania w zagadnieniach brzegowych, Wydawnictwo Pracownia Komputerowa Jacka Skalmierskiego, Gliwice, 1999.

[8] Blahut R., Theory and Practice of Error Control Codes, Addison-Wesley Publishing Company, Massachusetts, USA, 1983.

[9] Bogucka H., Dziech A., Sawicki J., Elementy cyfrowego przetwarzania sygnałów z przykładami zastosowań i wykorzystaniem środowiska MATLAB, Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków, 1999.

[10] Brzózka J., Dobraczyński L., Programowanie w Matlabie, Wyd. Mikom, Warszawa, 1998.

[11] Castleman K., Digital image processing, Prentice Hall, New Jersey, 1996.

[12] Cooley J., Tookey J., *An algorithm for the machine calculation of complex fourier series*, Mathematics of Computation, 19, 297–301, 1965.

[13] Drygajło A., Rumatowski K., Analiza sekwencyjnościowa układów liniowych, PWN, Warszawa, 1990.

[14] Harmuth H., Sequency Theory. Fundations and Applications, Academic Press Inc., New York, 1977.

[15] Hurst S., Miller D., Muzio J., Spectral Techniques in Digital Logic, Academic Press Inc., London, 1985.

[16] Kołodziej W., Wybrane rozdziały analizy matematycznej, PWN, Warszawa, 1970.

[17] Kulesza W., Systemy widmowej analizy danych cyfrowych, WKŁ, Warszawa, 1984.

[18] Łuba T., Synteza układów logicznych, Oficyna Wydawnicza PW, Warszawa, 2005.

[19] Łuba T., Jasiński K., Zbierzchowski B., Specjalizowane układy cyfrowe w strukturach PLD i FPGA, WKŁ, Warszawa, 1997.

[20] Mallat S., *Theory for multiresolution signal decomposition: The wavelet representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 674–693, 1989.

[21] Mostowski A., Stark M., Algebra liniowa, WNT, Warszawa, 1975.

[22] Mrozek B., Mrozek Z., Matlab – uniwersalne środowisko do obliczeń naukowo-technicznych, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1996.

[23] Porwik P., *The spectral test of the boolean function linearity*, Int. J. Appl. Math. Comput. Sci., 13(4), 567–575, 2003.

[24] Porwik P., Przekształcenia ortogonalne dla ekstrahowania cech widma danych binarnych, Wydawnictwo Uniwersytetu Śląskiego, Katowice, 2004.

[25] Porwik P., Lisowska A., *The haar wavelet transform in digital image processing: its status and achievements*, Int. Journal Machine Graphics and Vision, 13, 79–98, 2004.

[26] Porwik P., Zaczkowski P., Wrobel K., *Some practical remarks about binary decision diagram size reduction*, IEICE Int. Journal Electronics Express, 3, 51–57, 2006.

[27] Pratap R., Matlab 7 dla naukowców i inzynierów, PWN, Warszawa, 2007.

[28] Rao K., Ahmed N., Orthogonal Transforms for Digital Signal Processing, Springer-Verlag, New York, 1975.

[29] Ross K., Wright C., Matematyka dyskretna, PWN, Warszawa, 1996.

[30] Skarbek W., Multimedia. Algorytmy i standardy kompresji, Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1998.

[31] Skoczylas Z., Jurlewicz T., Algebra liniowa, Oficyna Wydawnicza GiS, Wrocław, 2000.

[32] Somenzi F., BDD Package: CUDD v.2.5.0, http://vlsi.colorado.edu/ fabio/CUDD/cuddIntro.html, 2012.

[33] Stanković R., Astola J., Spectral Interpretation of Decision Diagrams, Springer-Verlag, New York, 2003.

[34] Szabatin J., Podstawy teorii sygnałów, WKŁ, Warszawa, 2000.

[35] Tadeusiewicz R., Korohoda P., Komputerowa analiza i przetwarzanie obrazów, Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków, 1997.

[36] Turowicz A., Teoria macierzy, Wydawnictwa AGH, Kraków, 1985.

[37] Wang R., Introduction to Orthogonal Transforms with applications in data processing and analysis, Cambridge University Press, UK, 2012.

[38] Wojtaszczyk P., Teoria falek, WNT, Warszawa, 2000.

[39] Wróbel Z., Koprowski R., Praktyka przetwarzania obrazów w programie Matlab, Akademicka Oficyna Wydawnicza Exit, Warszawa, 2004.

[40] Zieliński T., Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań, WKŁ, Warszawa, 2005.

Piotr Porwik

Selected Methods of Digital Signal Processing with Programs in Matlab

S u m m a r y

The aim of the present monograph is to discuss the theory of digital signal processing and examples of algorithms describing both the theory and practical implementations. The proposed algorithms were implemented in a form of programs written in the Matlab language. The book consists of two parts. Part One presents the theory of spectral analysis of signals used herein, which is next illustrated by examples of spectral analysis and spectral synthesis algorithms. Part Two of the monograph is devoted to the presentation of algorithms utilized in the process of implementing the discussed theory. First, theoretical bases for processing discrete signals are outlined. Then, the readers are provided with definitions and the mathematical derivations which facilitate the understanding of all the functional algorithms described in the monograph. Part Two contains an overview of various implementations of the previously discussed theory in the areas of image processing and spectral analysis of Boolean functions, by proving the respective theorems. The theorems should facilitate the formulation of corresponding algorithms for image and Boolean functions analysis.

Ausgewählte Methoden der digitalen Signalverarbeitung mit Programmen in Matlab

Z u s a m m e n f a s s u n g

Ziel der hier vorliegenden Monographie ist die Erörterung der Theorie der digitalen Signalverarbeitung sowie der Beispielalgorithmen die sowohl die Theorie als auch die praktische Durchführung beschreiben. Die vorgeschlagenen Algorithmen wurden implementiert in Form von Programmen, die in der Programmiersprache Matlab erstellt wurden. Das Buch setzt sich aus zwei Teilen zusammen. Der erste Teil stellt die Theorie der Spektralanalyse der verwendeten Signale vor, welche wiederum durch Beispiele von Spektralanalysen und spektralen Synthesealgorithmen veranschaulicht werden. Im zweiten Teil der Monographie werden die Algorithmen präsentiert, die im Implementierungsprozess der zuvor erörterten Theorie verwendet wurden. Zuerst werden die theoretischen Grundlagen für die Verarbeitung diskreter Signale beschrieben. Dann werden den Lesern Definitionen und mathematische Ableitungen bereitgestellt, die das Verständnis für die in dieser Monographie beschriebenen Algorithmen erleichtern. Der zweite Teil enthält einen Überblick über verschiedenen Implementierungen der zuvor diskutierten Theorie auf dem Gebiet der Bildverarbeitung und Spektralanalyse von Booleschen Funktionen, wobei die entsprechenden Theoreme nachgewiesen werden. Die Theoreme sollten die Formulierung von entsprechenden Algorithmen zur Bild- und Booleschen Funktionsanalyse erleichtern.

Redaktor: Barbara Todos-Burny
Projektant okładki: Magdalena Starzyk
Redaktor techniczny: Barbara Arenhövel

ISSN 0208-6336

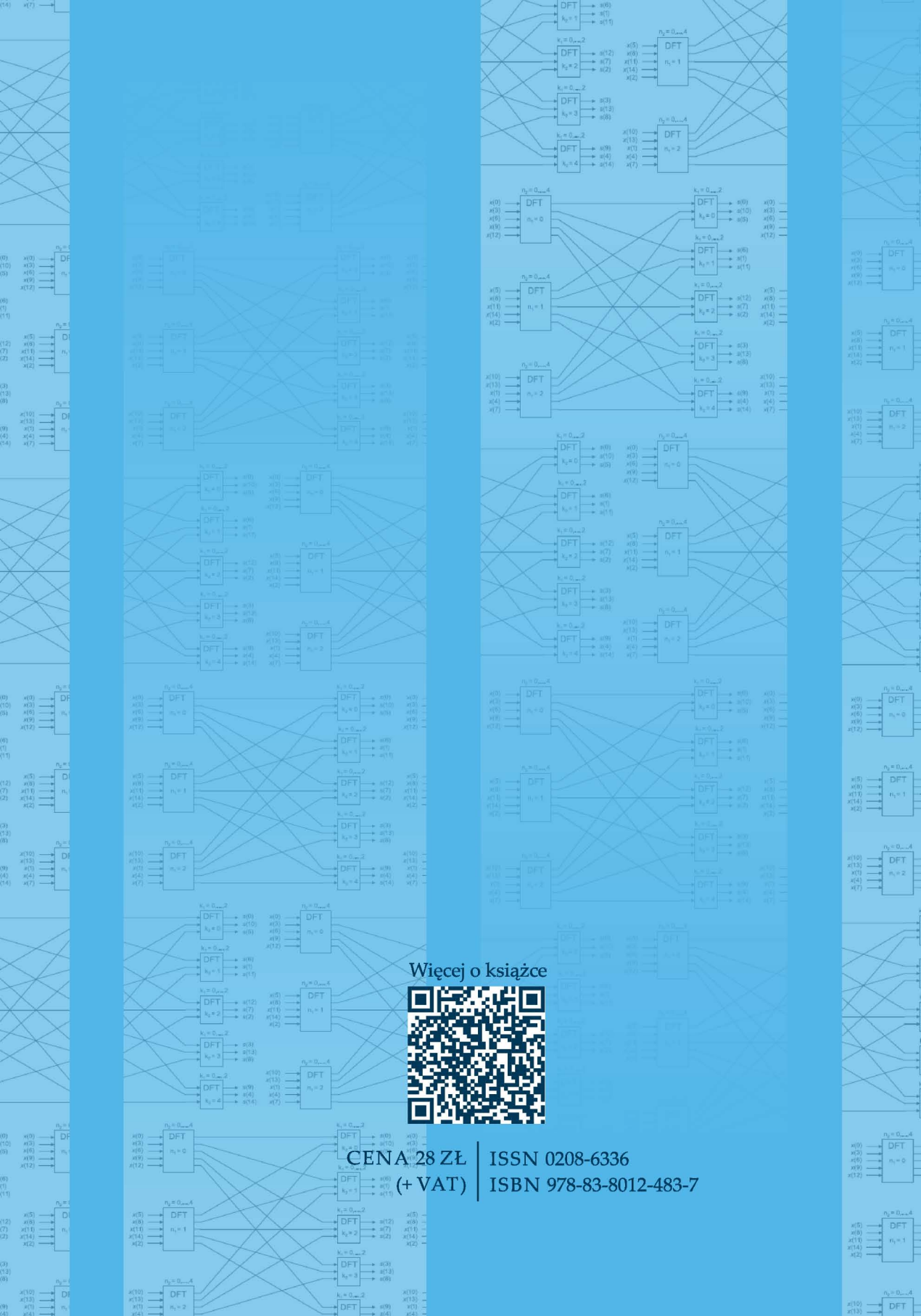
ISBN 978-83-8012-483-7
(wersja drukowana)

ISBN 978-83-8012-484-4
(wersja elektroniczna)

Wydawca
Wydawnictwo Uniwersytetu Śląskiego
ul. Bankowa 12B, 40-007 Katowice
www.wydawnictwo.us.edu.pl
e-mail: wydawus@us.edu.pl

Wydanie I. Ark. druk. 14,0. Ark. wyd. 13,0.
Papier offset. kl. III, 90 g Cena 28 zł (+ VAT)

Druk i oprawa: EXPOL P. Rybiński, J. Dąbek Spółka Jawna
ul. Brzeska 4, 87-800 Włocławek



Więcej o książce

CENA 28 ZŁ (+ VAT)
ISSN 0208-6336
ISBN 978-83-8012-483-7