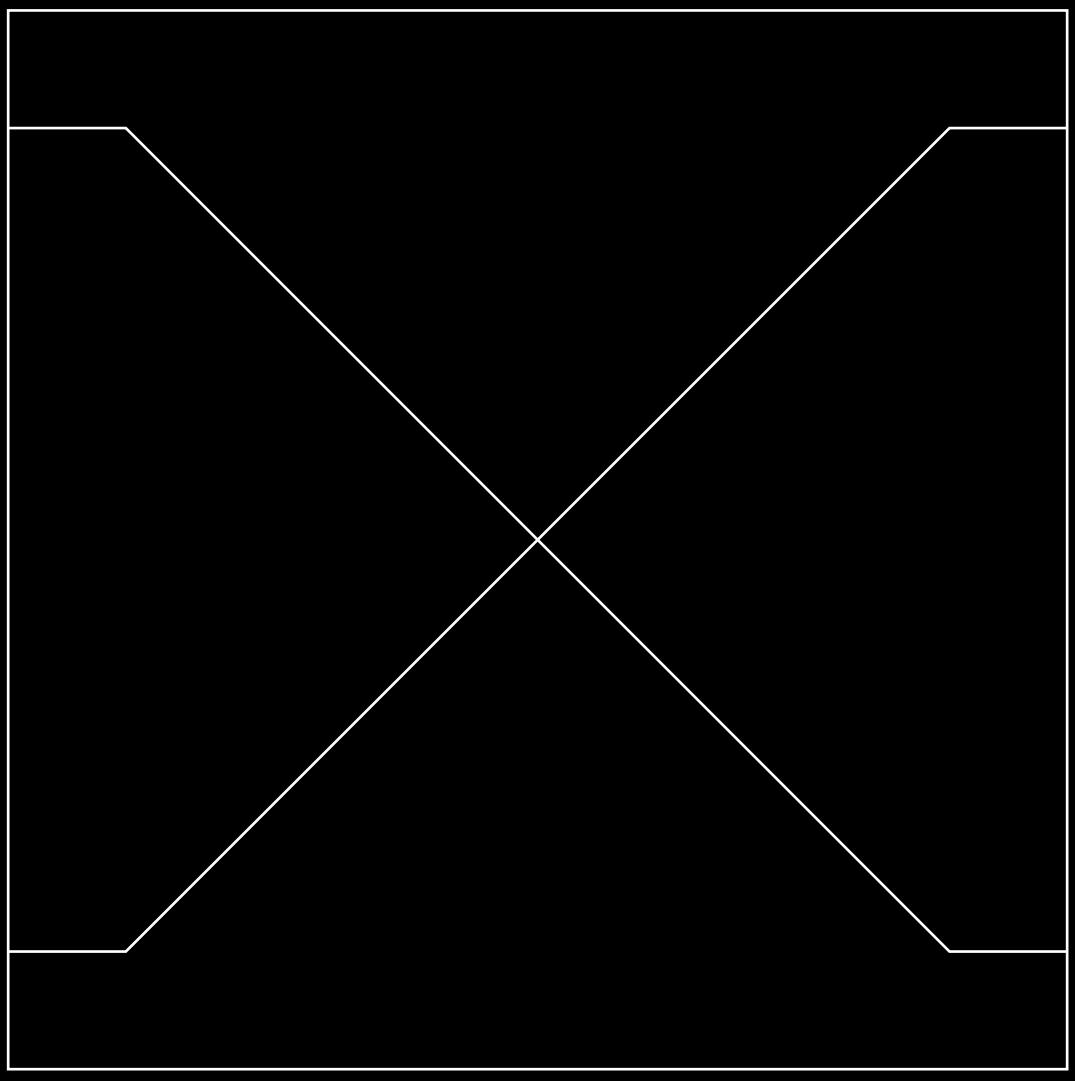
Monografie Politechniki Łódzkiej

Szybkie algorytmy adaptacyjne przekształceń trygonometrycznych

Dariusz Puchała



Dariusz Puchała

Szybkie algorytmy adaptacyjne przekształceń trygonometrycznych Łódź 2016

Szybkie algorytmy adaptacyjne przekształceń trygonometrycznych

Dariusz Puchała

Monografie Politechniki Łódzkiej

Łódź 2016

Recenzenci:
dr hab. inż. Maria Pietruszka, prof. Politechniki Łódzkiej
dr hab. inż. Piotr Lipiński

Redaktor Naukowy Wydziału Fizyki Technicznej,
Informatyki i Matematyki Stosowanej:
dr hab. inż. Aneta Poniszewska-Marańda

Projekt okładki:
dr inż. Krzysztof Guzek

WYDAWNICTWO POLITECHNIKI ŁÓDZKIEJ
90-924 Łódź, ul. Wólczańska 223
tel. 42-631-20-87, 42-631-29-52
fax 42-631-25-38
e-mail: zamowienia@info.p.lodz.pl
www.wydawnictwa.p.lodz.pl

ISBN 978-83-7283-886-5

Reprodukcja z materiałów dostarczonych przez Autorów

Nakład 50 egz. Ark. druk. 15,0. Papier offset. 80 g, 70 x 100
Wykonano w Drukarni „Quick-Druk" s.c. 90-562 Łódź, ul. Łąkowa 11
Nr 2241

Książkę tę dedykuję mojej żonie
Annie

Spis treści

1

Wprowadzenie

Na treść niniejszej monografii składa się opis szybkich algorytmów adaptacyjnych dla numerycznego obliczania przekształcenia Fouriera w postaci całkowej. Jako kwadratury całkowania numerycznego wykorzystano znane dyskretne przekształcenia trygonometryczne w postaci dyskretnego przekształcenia Fouriera oraz dyskretnych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju. Przedstawione szybkie algorytmy adaptacyjne dają możliwość obliczania zadanego pasma widma sygnału zgodnie z kryterium: *dokładność-czas realizacji obliczeń.*

W ramach monografii rozważa się jednocześnie przypadki sygnałów jedno- i dwuwymiarowych. Proponowane podejście może jednak zostać z powodzeniem rozszerzone na większą liczbę wymiarów.

1.1. Przegląd zastosowań przekształceń trygonometrycznych

Nazwa przekształcenia Fouriera[†] wywodzi się od nazwiska francuskiego matematyka J. B. J. Fouriera, który prowadząc prace nad rozwiązaniem równania ciepła w roku 1807 udowodnił, iż dla szerokiego zakresu funkcji możliwe jest ich przedstawienie w postaci zbieżnego szeregu okresowych funkcji sinus i kosinus. Wyniki badań Fouriera były zwieńczeniem prac innych współczesnych mu matematyków, którzy w różnych aspektach badali zagadnienia reprezentacji arbitralnych funkcji w postaci skończonych i nieskończonych szeregów okresowych funkcji trygonometrycznych. Byli to między innymi: L. Euler - analiza nieskończonych szeregów funkcji kosinus; A. C. Clairaut - wyznaczanie kształtów orbit planet w oparciu o skończony zbiór obserwacji, podał on także pierwszą formułę dyskretnego przekształcenia Fouriera[†] (r. 1754); L. Lagrange - prowadził prace nad skończonymi szeregami funkcji sinus; D. Bernoulli - przedstawił rozwiązanie zagadnienia drgającej struny w oparciu o nieskończone szeregi funkcji sinus i kosinus (r. 1753) [48, 133].

Obecnie przekształcenie Fouriera odgrywa ważną rolę w wielu gałęziach nauki. Po pierwsze, jako przekształcenie liniowe, jest pomocne w rozwiązywaniu wielu zagadnień

[†]Definicje całkowego oraz dyskretnego przekształcenia Fouriera zamieszczone zostały w rozdziale 2 niniejszej pracy.

dotyczących układów o charakterze liniowym. Po drugie pozwala na przedstawienie sygnału wejściowego względem bazy ciągłych funkcji sinusoidalnych, które sparametryzowane są zmienną rzeczywistą. Zmienna ta posiada silną interpretację fizyczną w postaci częstotliwości. Stąd przekształcenie Fouriera daje możliwość pomiaru widm częstotliwościowych sygnałów i tym samym pozwala na badanie zjawisk o charakterze falowym, między innymi z zakresu: optyki, akustyki (taki sposób przedstawienia sygnałów akustycznych jest jak najbardziej zgodny ze sposobem postrzegania dźwięku przez człowieka), fizyki kwantowej, elektroniki i elektrotechniki, a także probabilistyki i zagadnień rozwiązywania cząstkowych równań różniczkowych [9, 14, 42, 94, 136].

W przypadku sygnałów wejściowych podawanych w postaci dyskretnej, tzn. jako skończone zbiory obserwacji dokonywanych w dyskretnych chwilach czasowych, do obliczania widm częstotliwościowych wykorzystuje się podejście numeryczne oparte o dyskretne przekształcenia trygonometryczne, takie jak dyskretne przekształcenie Fouriera, a także dyskretne przekształcenia kosinusowe i sinusowe drugiego i czwartego rodzaju. Dzięki temu przekształcenie Fouriera może być stosowane w zadaniach cyfrowego przetwarzania danych, w których stanowi narzędzie o dużym potencjale i możliwościach. Jego zastosowania dotyczą między innymi takich zagadnień jak: analiza widmowa i filtracja sygnałów, szybkie obliczanie splotu i funkcji korelacji sygnałów, analiza układów liniowych, kompresja i kodowanie danych, etc. [34, 73, 161].

Współczesne zastosowania dyskretnego przekształcenia Fouriera sięgają daleko poza dyscypliny o charakterze matematyczno-technicznym. Przekształcenie to znajduje zastosowania między innymi w takich dziedzinach nauki jak: medycyna, biologia, kryminalistyka i medycyna sądowa, topografia, biometria, czy też informatyka. Jako próbę prezentacji przykładowych zastosowań dyskretnego przekształcenia Fouriera można podać następujące zagadnienia: opracowanie biologicznego modelu metabolizmu ciał ketonowych [33], zwiększanie współczynnika wykrywalności obiektów w Dopplerowskich systemach radarowych [17], analiza przebiegu napięć w trakcie wyładowań elektrostatycznych ładunków gromadzonych na ludzkim ciele [72], zwiększenie czułości testów w badaniu diagnostycznym słuchu za pomocą metody potencjałów słuchowych wywołanych [148], konstrukcja kodów cyklicznych dla zadań kodowania informacji [142], redukcja poziomu szumów w ultradźwiękowej tomografii dyfrakcyjnej [15], szyfrowanie sygnałów mowy [36], opracowanie metody diagnozowania stenozy mitralnej oraz stenozy zastawki trójdzielnej w oparciu o ultrasonografię dopplerowską [59], klasyfikacja i rozpoznawanie obiektów w oparciu o ich kształty [65, 138], symulacja i analiza przepięć w układach prądu stałego [80], rozpoznawanie twarzy [115, 116], obliczanie parametrów pól elektromagnetycznych o wielu źródłach [96], rekonstrukcja obrazów w systemach radarowych z syntetyczną aperturą [43], analiza aktywności mózgu na podstawie danych elektroencefalograficznych [37], badanie charakterystyk światłowodów budowanych z wykorzystaniem kryształu fotonicznego [89], czy też zagadnienia projektowania współpłaszczyznowych zespołów anten [61].

Jak już wspomniano, do numerycznego obliczania przekształcenia Fouriera wykorzystuje się, oprócz dyskretnego przekształcenia Fouriera, także dyskretne przekształcenia kosinusowe i sinusowe drugiego i czwartego rodzaju[†]. Przekształcenia te są matematyczną podstawą wielu metod dotyczących obszaru informatyki. Znajdują szerokie zastosowania w zadaniach cyfrowego przetwarzania obrazów i dźwięku, w filtracji sy-

[†]Definicje dyskretnych przekształceń kosinusowych i sinusowych zamieszczono w rozdziale 2 niniejszej pracy

gnałów, w algorytmach rozpoznawania wzorców i ekstrakcji cech, przy rekonstrukcji obrazów, a także w stratnej kompresji danych, etc. [5, 110, 124, 118]. W szczególności dyskretne przekształcenie kosinusowe drugiego rodzaju jest przekształceniem najczęściej wykorzystywanym w zadaniach stratnej kompresji sygnałów. Jego podstawową zaletą jest dobre dopasowanie funkcji bazowych do sygnałów nieciągłych na okręgu oraz własność koncentracji energii w małej liczbie współczynników niskoczęstotliwościowych dla sygnałów o wysokiej korelacji. W rzeczywistości przekształcenie to dla sygnałów modelowanych jako sygnały Markowa pierwszego rzędu, przy współczynniku autokorelacji sygnału bliskiemu jedności, będzie zbliżać się asymptotycznie do optymalnego w sensie koncentracji energii przekształcenia Karhunena-Loévego [118, 124]. Własność ta jest równie cenna w zadaniach klasyfikacji i rozpoznawania wzorców, ponieważ pozwala na konstruowanie krótkich wektorów cech, przy jednoczesnym zapewnieniu wysokich trafności klasyfikacji. Jako przykłady zastosowań dyskretnego przekształcenia kosinusowego drugiego rodzaju można podać: stratną kompresję obrazów statycznych JPEG [118, 124, 127], stratną kompresję sekwencji obrazów ruchomych i dźwięku: MPEG-1, MPEG-2 [118, 124, 128, 129], oraz standardy H.261 [118, 124, 130] i H.263 [118, 124, 131]; segmentację obrazów [58, 102], szeregowanie tekstów według kryterium częstości występowania zadanej frazy [93], kategoryzacja tekstów [97], zadania rozpoznawania twarzy [45, 57, 71, 117], kompensacja wpływu zmiennego oświetlenia w systemach rozpoznawania twarzy [21], wyszukiwanie sekwencji wideo o zbliżonej treści wizualnej w bibliotekach klipów multimedialnych [63], wydobywanie sygnałów mowy spod wpływu hałaśliwego tła otoczenia [20], techniki wykrywania zwolnień tempa bicia serca płodu [146], a także rekonstrukcja obrazów podpróbkowanych [1] oraz klasyfikacja tekstur [68, 123] i wiele innych.

Pozostałe przekształcenia dyskretne, tj. przekształcenie kosinusowe czwartego rodzaju, oraz sinusowe przekształcenia rodzaju drugiego i czwartego wykorzystuje się między innymi do: kompresji dźwięku w standardach MP3, AC3 [124, 160] (dyskretne przekształcenie kosinusowe czwartego rodzaju), rozpoznawania głosek [83] (dyskretne przekształcenie sinusowe drugiego rodzaju), budowy przekształceń zakładkowych LOT [75] (dyskretne przekształcenie sinusowe czwartego rodzaju), obliczania równań różniczkowych oraz implementacji filtrów o skończonej odpowiedzi impulsowej.

Dla sygnału opisanego poprzez zbiór N próbek złożoność procesu obliczania przekształcenia dyskretnego jest rzędu $\mathcal{O}(N^2)$ i wynika z konieczności wymnożenia N elementowego wektora poprzez macierz kwadratową o N na N elementach. Taka ilość obliczeń, nawet pomimo szybkiego rozwoju maszyn cyfrowych, stanowiła do połowy dwudziestego wieku poważną barierę dla praktycznego wykorzystania przekształcenia Fouriera w zadaniach cyfrowej analizy i przetwarzania danych. Choć w literaturze odnotowuje się kilka wcześniejszych prac o równoważnych wynikach dotyczących redukcji złożoności obliczeniowej dyskretnego przekształcenia Fouriera, są to dla przykładu prace takich autorów jak: C. F. Gauss (r. 1805), C. Runge i H. Konig (r. 1924), G. C. Danielson i C. Lanczos (r. 1942) [27, 48], to jako datę przełomową podaję się rok 1965, w którym dzięki pracy [28] spopularyzowany został szybki algorytm o złożoności obliczeniowej $\mathcal{O}(Nlog_2N)$. Taką redukcję obliczeń osiągnięto dzięki dostrzeżeniu pewnych związków symetrii w macierzy opisującej dyskretne przekształcenie Fouriera. Od tamtej chwili wielu autorów podejmowało prace nad dalszym ulepszaniem istniejących, oraz poszukiwaniem nowych algorytmów, bardziej efektywnych obliczeniowo[†], w tym także

[†]Skrótowy przegląd szybkich algorytmów obliczania dyskretnego przekształcenia Fouriera zamiesz-

realizacji sprzętowych, zarówno analogowych, dla przykładu: realizacje szybkich algorytmów w oparciu o wzmacniacze operacyjne [41, 108], projekt analogowego procesora sygnałowego dla szybkiego algorytmu Fouriera budowanego w technice tranzystorów z pływającą bramką [64], siatka elementów LC imitujących optyczne zjawiska dyfrakcji i refrakcji światła dla obliczania dyskretnego przekształcenia Fouriera [2]; jak i cyfrowych o różnorodnych architekturach: sekwencyjnych, kaskadowych [12] i równoległych, między innymi: budowanych w oparciu o procesory sygnałowe DSP [125] oraz z wykorzystaniem programowalnych układów logicznych typu FPGA [19, 95, 153], a także propozycje algorytmów dla architektur równoległych [29, 31, 79, 85, 140].

Podobnie dyskretne przekształcenia kosinusowe i sinusowe drugiego oraz czwartego rodzaju, które zostały odkryte w połowie lat siedemdziesiątych [5] (kosinusowe drugiego rodzaju) oraz w połowie lat osiemdziesiątych ubiegłego stulecia [144] (pozostałe przekształcenia), dzięki swej ogromnej popularności doczekały się do dnia dzisiejszego licznych implementacji programowych[‡] oraz sprzętowych. Jako przykłady implementacji sprzętowych można podać realizacje analogowe: w postaci tablicy przełączanych kondensatorów oraz układów całkujących [139], oraz realizacje cyfrowe: w postaci układów VLSI [23, 67, 70, 50], a także te, przeznaczone dla układów FPGA [78, 84, 92].

Rozwój nowoczesnych technologii multimedialnych (np. standard HDTV telewizji o rozdzielczościach obrazu rzędu 1920 na 1080 pikseli, czy też wielokanałowe systemy kodowania dźwięku: Dolby Digital, DTS, etc.), a także rosnące możliwości urządzeń służących do akwizycji danych obrazowych i akustycznych, pomimo zwiększanych mocy obliczeniowych współczesnych komputerów, pociągają za sobą ciągłą potrzebę udoskonalania istniejących oraz poszukiwania nowych efektywnych algorytmów obliczania wspomnianych przekształceń trygonometrycznych. Ma to szczególne znaczenie w systemach czasu rzeczywistego, np. w systemach automatycznego rozpoznawania otoczenia w robotyce (w jęz. ang. *machine vision*), czy też w przemysłowej kontroli jakości (w jęz. ang. *automated image analysis*) [77].

W niniejszej monografii opisana została propozycja jednego z takich udoskonaleń.

1.2. Układ książki

W rozdziale 2 zamieszczono definicje jedno- i dwuwymiarowych całkowych przekształceń Fouriera, oraz kosinusowych i sinusowych przekształceń Fouriera. Następnie pokazano zależności łączące wspomniane przekształcenia całkowe z dyskretnymi przekształceniami Fouriera oraz przekształceniami kosinusowymi i sinusowymi drugiego i czwartego rodzaju. Dodatkowo wymienione zostały (wraz z dowodami) wybrane własności symetrii w grupie rozważanych przekształceń dyskretnych, które posłużyły w rozdziale 4 do syntezy szybkich algorytmów z przerzedzeniem w czasie.

Rozdział 3 poświęcono zagadnieniom numerycznego obliczania całkowych przekształceń Fouriera. W rozdziale tym przekształcenia dyskretne rozważa się jako kwadratury (kubatury w przypadku sygnałów dwuwymiarowych) całkowania numerycznego. Stąd pierwszą część rozdziału poświęcono teorii całkowania numerycznego, w tym zasadom konstruowania kwadratur (kubatur) złożonych, oraz wyrażeniom oceny błędów przybliżonego obliczania całek. Następnie, w drugiej części rozdziału, przedstawio-

czono w rozdziale 4.

[‡]Skrótowy przegląd szybkich algorytmów obliczania dyskretnych przekształceń kosinusowych i sinusowych obu rozważanych rodzajów zamieszczono w rozdziale 4.

no propozycje algorytmów adaptacyjnych dla rozważanych dyskretnych przekształceń jedno- i dwuwymiarowych, wraz z proponowanymi heurystycznymi wyrażeniami oceny błędów numerycznego obliczania przekształceń całkowych, które to wyrażenia skonstruowano w metryce Czebyszewa.

Do budowy szybkich algorytmów adaptacyjnych wymagane są szybkie algorytmy z przerzedzeniem w czasie. Stąd w rozdziale 4 Czytelnik znajdzie wzory rozkładów takich algorytmów dla wszystkich rozważanych przekształceń dyskretnych. W przypadku dyskretnych jedno- i dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju, poza wzorami rozkładu, pokazano również sposoby ich wyprowadzeń na przykładzie dyskretnego przekształcenia kosinusowego drugiego rodzaju.

W rozdziale 5 zamieszczono propozycje szybkich algorytmów adaptacyjnego obliczania przekształceń całkowych. Ponadto przeprowadzono ponowną dyskusję heurystycznych wyrażeń oceny błędów, co wymuszała zmiana schematu doboru próbek sygnału dla szybkich algorytmów obliczania dyskretnych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju.

Wyrażenia oceny błędów numerycznego obliczania całek rozszerzono w rozdziale 6 na inne metryki popularne w praktyce cyfrowego przetwarzania sygnałów, tj. względne i bezwzględne metryki błędu średniokwadratowego oraz stosunku szczytowego sygnału do szumu, a także względną metrykę Czebyszewa.

W rozdziale 7 zamieszczono wyniki eksperymentalnej weryfikacji skuteczności proponowanych wyrażeń oceny błędów. Badania przeprowadzono z wykorzystaniem kilku sygnałów modelowych o znanej postaci analitycznej. Badano także wpływ dodatkowych operacji arytmetycznych związanych z oceną błędu na całkowity czas realizacji obliczeń.

W rozdziale 8 podano przykład praktycznego zastosowania szybkiego algorytmu adaptacyjnego dla dyskretnego przekształcenia Fouriera w zadaniach klasyfikacji obiektów realizowanej w oparciu o deskryptory fourierowskie, które obliczano dla konturów obiektów.

W niniejszej monografii zestawiono wyniki badań naukowych prowadzonych w ramach pracy doktorskiej realizowanej w Instytucie Informatyki Politechniki Łódzkiej pod kierunkiem Prof. zw. dr. hab. Michała Jacymirskiego. W tym miejscu pragnę złożyć serdeczne podziękowania Panu Prof. zw. dr. hab. Michałowi Jacymirskiemu za nieocenioną pomoc udzieloną mi w trakcie realizacji samych badań, jak również przy przygotowywaniu pracy doktorskiej, a także za wyrozumiałość i motywację do krytycznego spojrzenia na rozważaną problematykę badawczą. Szczególne podziękowania pragnę złożyć Panu Profesorowi za inspirację do samodzielnego zgłębiania zagadnień naukowych, oraz za wskazanie tego, jak wielce pasjonującym zadaniem może być samodzielne rozwiązywanie problemów badawczych.

2

Definicje i podstawowe własności przekształceń trygonometrycznych

W rozdziale "Definicje i podstawowe własności przekształceń trygonometrycznych" zamieszczone zostały definicje par całkowych jedno- i dwuwymiarowych przekształceń trygonometrycznych: Fouriera oraz kosinusowego i sinusowego przekształcenia Fouriera. Krótkiej charakterystyce poddano także warunki istnienia oraz wzajemnej jednoznaczności wspomnianych przekształceń. Następnie poprzez wprowadzenie odpowiednich reguł dyskretyzacji w dziedzinie czasu oraz w dziedzinie częstotliwości wykazano ścisłe związki pomiędzy całkowymi przekształceniami Fouriera, a dyskretnym przekształceniem Fouriera oraz dyskretnymi kosinusowymi i sinusowymi przekształceniami drugiego i czwartego rodzaju. Związki te stanowić będą punkt wyjścia do rozważań o przybliżonym, numerycznym obliczaniu całkowych przekształceń trygonometrycznych, które to zagadnienia dokładnie omówiono w trzecim rozdziale niniejszej monografii.

Stąd kolejno w sekcjach 2.1 - 2.3 zamieszczono definicje par jednowymiarowych przekształceń trygonometrycznych Fouriera w postaci całkowej, a następnie poprzez wprowadzenie odpowiednich reguł dyskretyzacji otrzymano szukane zależności łączące ze sobą przekształcenia całkowe z przekształceniami dyskretnymi. Z kolei w sekcji 2.4 podano wraz z wyprowadzeniami podstawowe własności symetrii dla dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju. Własności te posłużą w rozdziale 4 do syntezy szybkich algorytmów obliczania wspomnianych dyskretnych przekształceń trygonometrycznych.

Ostatnią część niniejszego rozdziału (patrz sekcja 2.5) poświęcono przypadkowi dwuwymiarowych przekształceń Fouriera. Podobnie do przypadku jednowymiarowego zamieszczono tutaj definicje par dwuwymiarowych przekształceń Fouriera oraz kosinusowego i sinusowego przekształcenia Fouriera w postaci całkowej, a następnie podano odpowiednie związki występujące pomiędzy dwuwymiarowymi przekształceniami całkowymi i dyskretnymi, wraz z regułami dyskretyzacji potrzebnymi do ich wykazania. Na końcu rozdziału zamieszczono podstawowe własności symetrii rozważanych dwuwymiarowych przekształceń dyskretnych, które to własności w rozdziale 4 wymagane będą do syntezy szybkich algorytmów obliczania dwuwymiarowego dyskretnego prze-

kształcenia Fouriera oraz dwuwymiarowych dyskretnych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju.

2.1. Przekształcenie Fouriera

Niech $x(t)$ oznacza funkcję zmiennej rzeczywistej t, która może przyjmować wartości rzeczywiste i różne od zera jedynie na pewnym przedziale $[0, T]$ dla $T \in \mathcal{R}$. Wówczas *proste przekształcenie Fouriera* funkcji $x(t)$ definiuje następująca całka (patrz [9, 14, 42, 94, 133, 136]), rozumiana w sensie Riemanna, postaci

$$X(\omega) = \mathcal{F}\{x(t)\} \triangleq \int_0^T x(t)e^{-i\omega t}dt, \tag{2.1}$$

gdzie parametr $\omega \in \mathcal{R}$ jest pulsacją, wyrażenie $e^{-i\omega t}$ zgodnie ze wzorem Eulera definiujemy jako $e^{-i\omega t} \triangleq cos(\omega t) - isin(\omega t)$, natomiast $i = \sqrt{-1}$ to jednostka urojona. W wyniku tak zdefiniowanego przekształcenia funkcji $x(t)$ zostaje przyporządkowana zespolona funkcja $X(\omega)$ zmiennej rzeczywistej ω, zwana *całką Fouriera*, *transformatą Fouriera* lub *widmem Fouriera* funkcji $x(t)$. O ile transformowana funkcja przyjmuje wartości niezerowe jedynie na pewnym zadanym przedziale $[0, T]$, to warunkiem wystarczającym istnienia całki (2.1) jest ograniczoność tej funkcji, oraz posiadanie przez nią co najwyżej przeliczalnej liczby punktów nieciągłości (patrz [133]). Często cytowanym w literaturze warunkiem dostatecznym istnienia całki Fouriera jest również fizyczna realizowalność transformowanej funkcji $x(t)$ (patrz [14]), o której mówimy wówczas, gdy funkcja reprezentuje pewne wielkości fizyczne.

Fundamentalnym twierdzeniem analizy fourierowskiej jest twierdzenie o *odwrotnym przekształceniu Fouriera* (patrz [9, 14, 42, 94, 133, 136]), zgodnie z którym funkcję $x(t)$ można wyrazić w postaci następującej całki z jej transformaty $X(\omega)$

$$x(t) = \mathcal{F}^{-1}\{X(\omega)\} \triangleq \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{i\omega t}d\omega \tag{2.2}$$

dla $t \in [0, T]$. Równość w powyższym wyrażeniu zachodzi jedynie w punktach, w których funkcja $x(t)$ jest ciągła. W punktach nieciągłości wyrażenie (2.2) jest prawdziwe tylko wtedy, gdy założymy, że funkcja $x(t)$ przyjmuje w nich następujące wartości

$$x(t) = \frac{x(t^+) + x(t^-)}{2}, \tag{2.3}$$

gdzie $x(t^-)$ i $x(t^+)$ to odpowiednio granica lewostronna i prawostronna funkcji $x(t)$ w punkcie $t \in [0, T]$. Wyrażenia (2.1) i (2.2) stanowią wówczas tzw. parę transformat Fouriera w postaci całkowej.

Wyprowadzenia wielu zależności prezentowanych w tym oraz w kolejnych rozdziałach niniejszej monografii oparte będą o podstawowe własności oraz twierdzenia dotyczące teorii przekształcenia Fouriera, których dokładne studium znaleźć można m. in. w następującej literaturze ([9, 14, 42, 94, 136]). Stąd w dalszej części monografii, wszędzie tam, gdzie w sposób bezpośredni korzystano z pewnych własności oraz twierdzeń z zakresu analizy fourierowskiej, zamieszczone zostały jedynie odpowiednie objaśnienia oraz odnośniki do właściwej literatury tematu.

2.2. Kosinusowe i sinusowe przekształcenie Fouriera

Arbitralnie wybraną funkcję $x(t)$, określoną na przedziale $[-T, T]$, można przedstawić w postaci sumy $x(t) = x_e(t) + x_o(t)$ jej składowych: parzystej $x_e(t)$ i nieparzystej $x_o(t)$ (patrz [94]), definiowanych odpowiednio jako

$$x_e(t) = (x(t) + x(-t))/2,$$

$$x_o(t) = (x(t) - x(-t))/2.$$

W chwili, gdy funkcja $x(t)$ jest *przyczynową funkcją czasu*, tzn. jest tożsamościowo równa zeru dla $t < 0$, to prawdziwa jest następująca zależność $x(t) = 2x_e(t) = 2x_o(t)$ (patrz [94]). W rozważanym przypadku zakładamy dodatkowo skończony czas obserwacji funkcji, tzn. przyjmujemy, że funkcja $x(t)$ może osiągać wartości różne od zera wyłącznie dla $t \in [0, T]$. Wówczas funkcję $x(t)$ można przedstawić w kategoriach transformat Fouriera jej składowej parzystej $X^e(\omega)$ i nieparzystej $X^o(\omega)$ jako [94]

$$x(t) = \frac{2}{\pi}\int_0^{+\infty} X^e(\omega)cos(\omega t)d\omega = \frac{2i}{\pi}\int_0^{+\infty} X^o(\omega)sin(\omega t)d\omega$$

dla $t \in [0, T]$, gdzie

$$\begin{aligned} X^e(\omega) &= \int_{-T}^{T} x_e(t)e^{-i\omega t}dt = \int_0^T x(t)cos(\omega t)dt, \\ X^o(\omega) &= \int_{-T}^{T} x_o(t)e^{-i\omega t}dt = -i\int_0^T x(t)sin(\omega t)dt. \end{aligned}$$

Wprowadźmy dodatkowo oznaczenia $X^C(\omega) \equiv X^e(\omega)$ oraz $X^S(\omega) \equiv X^o(\omega)$. Zgodnie z powyższymi wyrażeniami otrzymujemy następujące dwie pary przekształceń Fouriera w postaci całkowej nazywane odpowiednio prostym i odwrotnym *kosinusowym przekształceniem Fouriera* (patrz [14, 94])

$$X^C(\omega) = \mathcal{F}_C\{x(t)\} \triangleq \int_0^T x(t)cos(\omega t)dt, \tag{2.4a}$$

$$x(t) = \mathcal{F}_C^{-1}\left\{X^C(\omega)\right\} \triangleq \frac{2}{\pi}\int_0^{+\infty} X^C(\omega)cos(\omega t)d\omega \tag{2.4b}$$

dla $t \in [0, T]$ oraz prostym i odwrotnym *sinusowym przekształceniem Fouriera* (patrz [14, 94]), definiowanymi jako

$$X^S(\omega) = \mathcal{F}_S\{x(t)\} \triangleq \int_0^T x(t)sin(\omega t)dt, \tag{2.5a}$$

$$x(t) = \mathcal{F}_S^{-1}\left\{X^S(\omega)\right\} \triangleq \frac{2}{\pi}\int_0^{+\infty} X^S(\omega)sin(\omega t)d\omega \tag{2.5b}$$

dla $t \in [0, T]$. Zatem definiowane zgodnie z powyższymi zależnościami całkowe kosinusowe i sinusowe przekształcenia Fouriera stanowią szczególne przypadki przekształcenia Fouriera w postaci całkowej, określonego dla przyczynowej funkcji czasu $x(t)$. Stąd dla par kosinusowych i sinusowych przekształceń Fouriera obowiązywać będą analogiczne warunki dotyczące ich istnienia oraz wzajemnej jednoznaczności (patrz wzór 2.3), jak w przypadku par całkowych przekształceń Fouriera definiowanych jako (2.1) i (2.2).

2.3. Związki pomiędzy całkowymi i dyskretnymi przekształceniami Fouriera

Współcześnie wiele problemów z zakresu analizy oraz przetwarzania danych realizuje się na cyfrowych procesorach sygnałowych oraz na komputerach w oparciu o przybliżone obliczenia numeryczne. Zatem praktyczne wykorzystanie maszyn cyfrowych wymaga konwersji sygnału z postaci ciągłej do postaci dyskretnego zbioru próbek. Jednakże posiadanie niepełnej informacji o sygnale $x(t)$, tzn. informacji zawartej w arbitralnie wybranym zbiorze próbek, w przypadku ogólnym pozwala jedynie na przybliżone obliczenie widma Fouriera sygnału $x(t)$.

W praktyce cyfrowego przetwarzania danych do obliczania widma fourierowskiego wykorzystuje się dyskretne przekształcenia trygonometryczne, między innymi dyskretne przekształcenie Fouriera oraz dyskretne kosinusowe i sinusowe przekształcenia drugiego i czwartego rodzaju. Ponieważ na treść niniejszej monografii składają się numeryczne metody obliczania przekształceń Fouriera, to dla dalszych rozważań kluczowe znaczenie będą mieć związki występujące pomiędzy przekształceniami Fouriera w postaci całkowej, a dyskretnymi przekształceniami trygonometrycznymi. W niniejszej sekcji dokładnie rozpatrzono wspomniane związki.

W tym celu zakładamy, że dana jest funkcja $x(t)$ zmiennej rzeczywistej t, przyjmująca wartości różne od zera jedynie na przedziale $[0, T]$, dla której istnieje całkowe przekształcenie Fouriera postaci (2.1). Ponieważ w świetle przyjętych założeń funkcja $x(t)$ jest tożsamościowo równa zeru dla parametru t przyjmującego wartości $t < 0$ oraz $t > T$, to zgodnie z *twierdzeniem o próbkowaniu*[†] oraz z zasadą dualizmu czasowo-częstotliwościowego przekształcenia Fouriera (patrz [9, 14, 42, 94, 136]), widmo $X(\omega)$ sygnału $x(t)$, liczone według wzoru (2.1), można jednoznacznie odtworzyć na podstawie dyskretnego zbioru jego próbek równoodległych o krok dyskretyzacji $\Delta\omega = 2\pi/T$. Przyjmijmy zatem następującą regułę dyskretyzacji w dziedzinie częstotliwości

$$\omega_k = k\Delta\omega$$

dla $k = 0, \pm 1, \pm 2, ..., \pm\infty$. Można wykazać, że po uwzględnieniu dyskretyzacji w dziedzinie częstotliwości, opisanej powyższą regułą, para przekształceń Fouriera (2.1) i (2.2) przyjmie postać następujących wyrażeń

$$X(k\Delta\omega) = \int_0^T x(t)e^{-ik\Delta\omega t}dt \tag{2.6a}$$

dla $k = 0, \pm 1, \pm 2, ..., \pm\infty$, oraz

$$x(t) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X(k\Delta\omega)e^{ik\Delta\omega t} \tag{2.6b}$$

dla $t \in [0, T]$.

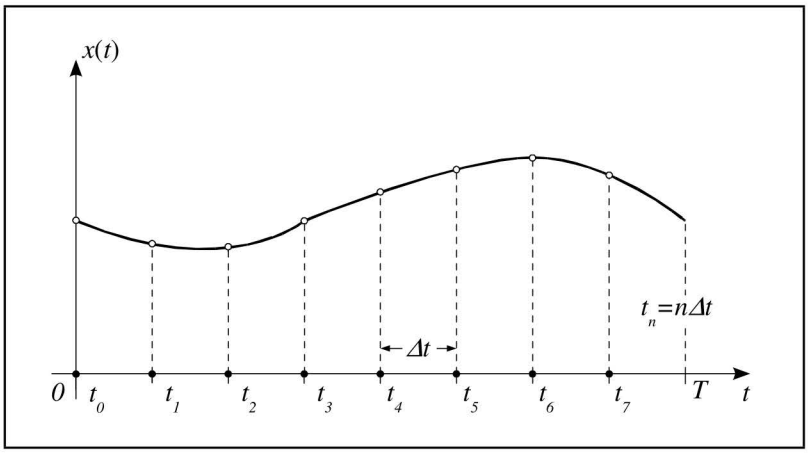
W praktyce cyfrowego przetwarzania danych transformowana funkcja $x(t)$ musi być reprezentowana poprzez skończony zbiór dyskretnych próbek, które pobierane są w pewnych odstępach czasu. W tym celu zakładamy, że na przedział $[0, T]$ przypada

[†] *Twierdzenie o próbkowaniu.*
Jeżeli widmo sygnału $x(t)$ jest równe zeru powyżej pewnej pulsacji ω_{max}, tzn. zachodzi $X(\omega) = 0$ dla $|\omega| > \omega_{max}$, to sygnał $x(t)$ jest jednoznacznie określony przez zbiór wartości $x(n\frac{\pi}{\omega_{max}})$, gdzie $n \in \mathcal{Z}$.

N próbek, gdzie N jest liczbą naturalną i różną od zera. Przyjmujemy ponadto, że próbki sygnału $x(t)$ pobierane są w punktach stanowiących początki podprzedziałów $[n\Delta t, (n+1)\Delta t]$ o jednakowych długościach równych $\Delta t = T/N$, przy czym parametr $n = 0, 1, ..., N-1$. Założoną dyskretyzację w czasie można wówczas opisać przy pomocy następującej reguły

$$t_n = n\Delta t$$

dla $n = 0, 1, ..., N-1$. Sposób doboru próbek realizowany według powyższego schematu został zobrazowany na rysunku 2.1. W rezultacie otrzymujemy dla sygnału wejściowego



Rysunek 2.1: Operacja próbkowania w dziedzinie czasu z doborem próbek z początków przedziałów o długościach $\Delta t = T/N$ dla przypadku $N = 8$ próbek

$x(t)$ skończony zbiór $\{x(n\Delta t) : n = 0, 1, ..., N-1\}$ jego dyskretnych próbek.

W przypadku ogólnym operacja próbkowania sygnału w czasie powoduje błąd obliczenia widma $X(\omega)$. Błąd ten, zgodnie z tw. o próbkowaniu, wynika bezpośrednio z faktu, iż widmo $X_p(\omega)$ sygnału spróbkowanego jest powielonym z całkowitymi wielokrotnościami $2\pi/\Delta t = N\Delta\omega$ (co odpowiada częstotliwości próbkowania $f_p = 1/\Delta t$) widmem $X(\omega)$, otrzymanym dla sygnału $x(t)$. W chwili, gdy widmo $X(\omega)$ jest ograniczone, tzn. istnieje taka wartość pulsacji $\omega_{max} \in \mathcal{R}$, że $X(\omega) = 0$ dla $|\omega| > \omega_{max}$ i ponadto $2\omega_{max} < N\Delta\omega$, to możliwe jest dokładne odtworzenie sygnału $x(t)$ na podstawie zbioru próbek $\{x(n\Delta t)\}$. W przeciwnym wypadku mamy do czynienia ze *zjawiskiem aliasingu* (patrz [25]), tj. z sytuacją, gdy powielone widma zachodzą na siebie. Zjawisko aliasingu pozwala na odtworzenie sygnału $x(t)$ z pewnym błędem zależnym od częstotliwości próbkowania. Stąd w przypadku ogólnym prawdziwa jest jedynie zależność $X(\omega_k) \approx X_p(\omega_k)$. Ponadto widmo sygnału spróbkowanego jest funkcją okresową o okresie równym $N\Delta\omega$, a zatem zachodzi również następująca równość postaci $X_p(\omega_{k+lN}) = X_p(\omega_k)$ dla dowolnego $l \in \mathcal{Z}$.

Po uwzględnieniu operacji próbkowania sygnału w dziedzinie czasu we wzorach (2.6a) i (2.6b), a także mając na względzie okresowość widma spróbkowanego sygnału, para przekształceń Fouriera przyjmuje teraz postać wyrażeń

$$X_p(k\Delta\omega) = \Delta t \sum_{n=0}^{N-1} x(n\Delta t) e^{-ik\Delta\omega n\Delta t} = \frac{T}{N} \sum_{n=0}^{N-1} x(n\Delta t) e^{-i\frac{2\pi}{N}kn} \qquad (2.7a)$$

dla $k = 0, 1, ..., N-1$, oraz

$$x(n\Delta t) = \frac{1}{T}\sum_{k=0}^{N-1} X_p(k\Delta\omega)e^{ik\Delta\omega n\Delta t} = \frac{1}{T}\sum_{k=0}^{N-1} X_p(k\Delta\omega)e^{i\frac{2\pi}{N}kn} \tag{2.7b}$$

dla $n = 0, 1, ..., N-1$. W tym miejscu należy zwrócić dodatkowo uwagę na pewną istotną własność symetrii funkcji $e^{-i\frac{2\pi}{N}kn}$, a mianowicie na własność, którą można zapisać w postaci $e^{-i\frac{2\pi}{N}(N-k)n} = e^{i\frac{2\pi}{N}kn} = e^{-i\frac{2\pi}{N}(-k)n}$. Stąd wynika natychmiast, iż

$$X_p(\omega_{-k}) = X_p(-k\Delta\omega) = X_p((N-k)\Delta\omega) = X_p(\omega_{N-k}) \tag{2.7c}$$

dla $k = 0, 1, ..., \frac{N}{2}$.

Zgodnie z pracami [4, 25, 73] dyskretne N-punktowe proste (DFT) i odwrotne (IDFT) przekształcenia Fouriera definiowane są jako

$$X_N(k) = DFT_N\{x(n)\} \triangleq \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-i\frac{2\pi}{N}kn} \tag{2.8a}$$

dla $k = 0, 1, ..., N-1$, oraz

$$x(n) = IDFT_N\{X_N(k)\} \triangleq \sum_{k=0}^{N-1} X(k)e^{i\frac{2\pi}{N}kn} \tag{2.8b}$$

dla $n = 0, 1, ..., N-1$. Porównując odpowiednio wyrażenia (2.7a) i (2.7b) ze wzorami (2.8a) i (2.8b), oraz mając na uwadze zależność postaci (2.7c), zauważymy, iż proste N-punktowe przekształcenie Fouriera (z dokładnością do stałej T) pozwala na obliczenie wartości widma sygnału spróbkowanego w punktach $\omega_k = k\Delta\omega$ dla $k = 0, \pm1, ..., \pm\frac{N}{2}$, operując na skończonym zbiorze jego próbek $\{x(n\Delta t) : n = 0, 1, ..., N-1\}$, które pozyskano z krokiem $\Delta t = T/N$. Natomiast dyskretne N-punktowe odwrotne przekształcenie Fouriera (z dokładnością do stałej $1/T$) umożliwia odtworzenie sygnału w punktach dyskretyzacji $t_n = n\Delta t$ dla parametru $n = 0, 1, ..., N-1$, na podstawie skończonego zbioru dyskretnych wartości widma $X_p(\omega_k)$ sygnału spróbkowanego, gdzie $k = 0, \pm1, ..., \pm\frac{N}{2}$. Można zatem zapisać, iż $X_p(\omega_k) = TX_N(k)$ dla $k = 0, \pm1, ..., \pm\frac{N}{2}$.

Poniżej zamieszczono wraz z dowodem istotną własność symetrii widma amplitudowego dyskretnego przekształcenia Fouriera, która jest prawdziwa dla sygnałów wejściowych przyjmujących wyłącznie wartości rzeczywiste.

Własność 2.3.1 *(Własność symetrii widma amplitudowego DFT dla sygnałów o wartościach rzeczywistych)*
Dla dyskretnego N-punktowego przekształcenia Fouriera, które operuje na dyskretnym zbiorze próbek $\{x(n)\}$, gdzie $x(n) \in \mathcal{R}$ dla $n = 0, 1, ..., N-1$, prawdziwa jest następująca równość $\|X_N(k)\| = \|X_N(N-k)\|$ dla $k = 0, 1, ..., \frac{N}{2}$, gdzie przez $\|\cdot\|$ rozumiemy moduł liczby zespolonej.

Dowód. Powyższą własność dowodzi się w sposób następujący. Zapiszmy

$$X_N(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)cos\left(\frac{2\pi}{N}kn\right) - i\frac{1}{N}\sum_{n=0}^{N-1} x(n)sin\left(\frac{2\pi}{N}kn\right)$$

dla $k = 0, 1, ..., \frac{N}{2}$, oraz

$$X_N(N-k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)cos\left(\frac{2\pi}{N}kn\right) + i\frac{1}{N}\sum_{n=0}^{N-1} x(n)sin\left(\frac{2\pi}{N}kn\right)$$

dla $k = 0, 1, ..., \frac{N}{2}$. Wówczas na mocy definicji modułu liczby zespolonej (patrz [73]) otrzymujemy natychmiast dowodzoną równość postaci $\|X_N(k)\| = \|X_N(N-k)\|$ dla parametru $k = 0, 1, ..., \frac{N}{2}$. ∎ *

Jak już wspomniano, w przypadku ogólnym dyskretyzacja w czasie powoduje błąd obliczenia widma sygnału $x(t)$ w punktach ω_k dla $k = 0, \pm 1, ..., \pm\frac{N}{2}$, który wynika ze zjawiska aliasingu. Zgodnie z pracą [136] moduł wartości tego błędu można wyznaczyć na podstawie następującej zależności

$$\epsilon(k) \triangleq \|X(\omega_k) - TX_N(k)\| = \left\| \sum_{\substack{l=-\infty \\ l\neq 0}}^{+\infty} X(\omega_k + \frac{2\pi}{\Delta t} l) \right\|$$

dla $k = 0, 1, ..., N-1$. Korzystając z *twierdzenia o zwartości widma*‡ (patrz [9, 14, 42, 94, 136]) można wykazać, że wartość tego błędu zależy wprost proporcjonalnie od kroku dyskretyzacji Δt. W przypadku ciągłej funkcji $x(t)$ oraz ciągłych $m-1$ jej pochodnych, wartość błędu $\epsilon(k)$ można oszacować według zależności

$$\epsilon(k) \leqslant \sum_{\substack{l=-\infty \\ l\neq 0}}^{+\infty} \frac{q}{\left|\omega_k + \frac{2\pi}{\Delta t} l\right|^{m+1}} \leqslant (\Delta t^2)^{m+1} \left(\sum_{l=1}^{+\infty} \frac{2q}{(2\pi T l)^{m+1}} \right),$$

gdzie $q \in \mathcal{R}$ oraz $q \geqslant 0$. Tym samym dobierając odpowiednio mały krok Δt (tzn. odpowiednio dużą wartość częstotliwości próbkowania $f_p = 1/\Delta t$) możemy zmniejszyć wartość błędu do poziomu akceptowalnego przy realizacji danego zadania. Zmniejszenie kroku dyskretyzacji automatycznie wiąże się ze wzrostem liczby próbek N, co powoduje wzrost liczby operacji arytmetycznych potrzebnych do obliczenia widma zgodnie ze wzorami (2.8a) oraz (2.8b). Stąd dobór odpowiedniej wartości kroku Δt jest jednym z kluczowych problemów cyfrowego przetwarzania danych.

Dotychczas pokazano, iż dyskretne N-punktowe przekształcenie Fouriera, operując na dyskretnym zbiorze próbek sygnału pozyskanych z krokiem dyskretyzacji Δt w punktach $t_n = n\Delta t$ dla $n = 0, 1, ..., N-1$, pozwala na przybliżone obliczenie widma sygnału $x(t)$ dla pulsacji ω_k, tzn. zachodzi wówczas związek $X(\omega_k) \approx TX_N(k)$ dla parametru $k = 0, \pm 1, ..., \pm\frac{N}{2}$.

W podobny sposób można wskazać związki pomiędzy całkowym kosinusowym (2.4a) oraz całkowym sinusowym (2.5a) przekształceniem Fouriera, a dyskretnymi N - punktowymi przekształceniami kosinusowymi i sinusowymi drugiego oraz czwartego rodzaju. W tym celu należy wprowadzić odpowiednio dobraną dyskretyzację w dziedzinie czasu oraz w dziedzinie częstotliwości. W obecnym przypadku przyjmujemy odmienną regułę dyskretyzacji sygnału $x(t)$ w czasie, polegającą na doborze próbek w środkach podprzedziałów $[n\Delta t, (n+1)\Delta t]$ długości $\Delta t = T/N$. Regułę tę można zapisać jako

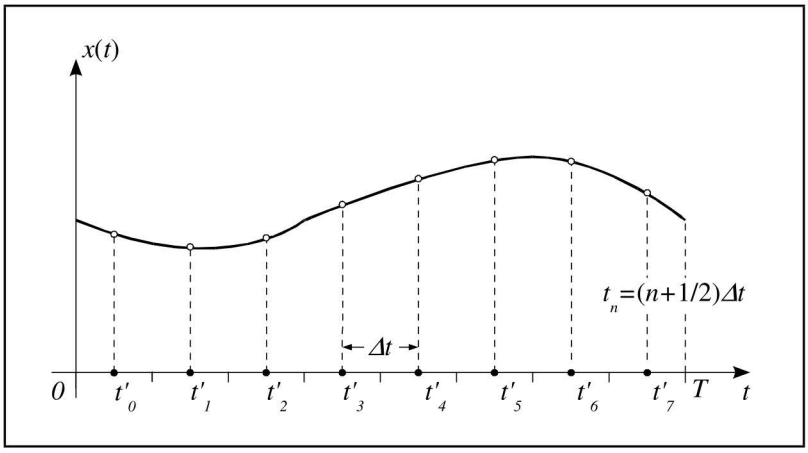
$$t'_n = (n + \frac{1}{2})\Delta t$$

* ∎ - q.e.d. (łac. Quod erat demonstrandum) "co było do udowodnienia", oznacza koniec dowodu.

‡ *Twierdzenie o zwartości widma.*
Jeżeli funkcja $x(t)$ i jej $n-1$ pochodnych jest ciągłych, to widmo takiej funkcji zanika przynajmniej tak szybko jak $|\omega|^{-(n+1)}$, co oznacza, że

$$\lim_{|\omega|\to\infty} |\omega|^n X(\omega) = 0.$$

dla $n = 0, 1, ..., N-1$. Sposób doboru próbek sygnału $x(t)$ realizowany zgodnie z powyższą regułą przedstawiono na rysunku 2.2.



Rysunek 2.2: Próbkowanie w czasie z doborem próbek w środkach przedziałów dyskretyzacji o długościach $\Delta t = T/N$ dla przypadku $N = 8$ próbek

Następnie w zależności od typu przekształcenia dyskretnego wprowadzamy odpowiednio zdefiniowaną dyskretyzację w dziedzinie częstotliwości. Dla przekształcenia kosinusowego i sinusowego drugiego rodzaju reguła ta przyjmuje postać

$$\omega_k^{II} = k\Delta\omega',$$

gdzie $\Delta\omega' = 2\pi/2T = \pi/T$, natomiast dla przypadku przekształceń czwartego rodzaju, dyskretyzację w częstotliwości definiujemy jako

$$\omega_k^{IV} = (k + \frac{1}{2})\Delta\omega'.$$

Po uwzględnieniu przedstawionych reguł dyskretyzacji w czasie i częstotliwości we wzorach (2.4a) oraz (2.5a) otrzymujemy wyrażenia pozwalające na przybliżone obliczenie widm $X^C(\omega)$ oraz $X^S(\omega)$ w punktach określonych regułami dyskretyzacji w dziedzinie częstotliwości i liczonych na bazie dyskretnych zbiorów próbek sygnału $x(t)$.

Dla reguł $t'_n = (n+\frac{1}{2})\Delta t$ i $\omega_k^{II} = k\Delta\omega'$ oraz całkowego przekształcenia kosinusowego Fouriera (2.4a) otrzymujemy wówczas

$$\begin{aligned} X_p^C(k\Delta\omega') &= \Delta t \sum_{n=0}^{N-1} x((n+\frac{1}{2})\Delta t) cos(k\Delta\omega'(n+\frac{1}{2})\Delta t) \\ &= \frac{T}{N} \sum_{n=0}^{N-1} x((n+\frac{1}{2})\Delta t) cos(\frac{\pi}{N} k(n+\frac{1}{2})) \end{aligned} \tag{2.9}$$

dla $k = 0, 1, ..., N-1$, gdzie $X_p^C(k\Delta\omega')$ oznacza widmo obliczone na podstawie dyskretnego zbioru próbek $\left\{x((n+\frac{1}{2})\Delta t) : n = 0, 1, ..., N-1\right\}$. Oczywiście przez wzgląd

na zjawisko aliasingu wynikające z dyskretyzacji w dziedzinie czasu, w przypadku ogólnym zachodzi następujący związek $X^C(k\Delta\omega') \approx X_p^C(k\Delta\omega')$ dla $k = 0, 1, ..., N-1$.

Zgodnie z pracą [4] dyskretne N-punktowe przekształcenie kosinusowe drugiego rodzaju (DCT-II) definiuje się jako

$$X_N^{CII}(k) = DCTII_N\{x(n)\} \triangleq \frac{p_k}{N}\sum_{n=0}^{N-1} x(n)cos(\frac{\pi}{N}k(n+\frac{1}{2})), \tag{2.10}$$

gdzie $k = 0, 1, ..., N-1$ oraz

$$p_k = \begin{cases} 2 \text{ dla } k = 0, \\ \\ 1 \text{ dla } k \neq 0. \end{cases}$$

Porównując wyrażenia (2.9) oraz (2.10) zauważymy, iż dyskretne N-punktowe przekształcenie kosinusowe drugiego rodzaju (z dokładnością do stałej T/p_k) umożliwia przybliżone obliczanie widma $X^C(\omega)$ w punktach $\omega_k^{II} = k\Delta\omega'$, operując na dyskretnym zbiorze próbek $\left\{x((n+\frac{1}{2})\Delta t) : n = 0, 1, ..., N-1\right\}$. Można zatem zapisać, iż dla tych przekształceń zachodzi związek postaci $X^C(k\Delta\omega') \approx T/p_k X_N^{CII}(k)$ dla następujących wartości parametru $k = 0, 1, ..., N-1$.

Z kolei przyjmując jako reguły dyskretyzacji w obu dziedzinach następujące wyrażenia $t'_n = (n+\frac{1}{2})\Delta t$ oraz $\omega_k^{IV} = (k+\frac{1}{2})\Delta\omega'$ i podstawiając je do wzoru (2.4a) definiującego całkowe przekształcenie kosinusowe Fouriera, tzn.

$$\begin{aligned} X_p^C((k+\frac{1}{2})\Delta\omega') &= \Delta t\sum_{n=0}^{N-1} x((n+\frac{1}{2})\Delta t)cos((k+\frac{1}{2})\Delta\omega'(n+\frac{1}{2})\Delta t) \\ &= \frac{T}{N}\sum_{n=0}^{N-1} x((n+\frac{1}{2})\Delta t)cos(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2})), \end{aligned} \tag{2.11}$$

otrzymujemy wyrażenie pozwalające na przybliżone obliczenie widma całkowego kosinusowego przekształcenia Fouriera w punktach $\omega_k^{IV} = (k+\frac{1}{2})\Delta\omega'$, liczonego na podstawie próbek $\left\{x((n+\frac{1}{2})\Delta t) : n = 0, 1, ..., N-1\right\}$. Oczywiście w przypadku ogólnym znów zachodzi związek $X^C((k+\frac{1}{2})\Delta\omega') \approx X_p^C((k+\frac{1}{2})\Delta\omega')$. Wyrażenie to z dokładnością do stałej T stanowi dyskretne N-punktowe przekształcenie kosinusowe czwartego rodzaju (DCT-IV), definiowane zgodnie z pracą [4] jako

$$X_N^{CIV}(k) = DCTIV_N\{x(n)\} \triangleq \frac{1}{N}\sum_{n=0}^{N-1} x(n)cos(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2})) \tag{2.12}$$

dla $k = 0, 1, ..., N-1$. Zatem dyskretne N-punktowe przekształcenie kosinusowe czwartego rodzaju pozwala na przybliżone obliczenie widma całkowego przekształcenia kosinusowego w punktach ω_k^{IV}, tzn. $X^C((k+\frac{1}{2})\Delta\omega') \approx TX_N^{CIV}(k)$ dla $k = 0, 1, ..., N-1$.

Stosując zbliżony do przedstawionego powyżej tok rozumowania można wskazać analogiczne związki postaci

$$X^S((k+1)\Delta\omega') \approx \frac{T}{p_k}X_N^{SII}(k)$$

oraz

$$X^S((k+\frac{1}{2})\Delta\omega') \approx TX_N^{SIV}(k)$$

dla $k = 0, 1, ..., N-1$ pomiędzy całkowym przekształceniem sinusowym określonym wzorem (2.5a), a dyskretnymi N-punktowymi przekształceniami sinusowymi drugiego (DST-II) i czwartego (DST-IV) rodzaju, definiowanymi według [4] odpowiednio jako

$$X_N^{SII}(k) = DSTII_N\{x(n)\} \triangleq \frac{p_k}{N}\sum_{n=0}^{N-1} x(n)sin(\frac{\pi}{N}(k+1)(n+\frac{1}{2})) \tag{2.13}$$

dla $k = 0, 1, ..., N-1$, oraz

$$X_N^{SIV}(k) = DSTIV_N\{x(n)\} \triangleq \frac{1}{N}\sum_{n=0}^{N-1} x(n)sin(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2})) \tag{2.14}$$

dla $k = 0, 1, ..., N-1$.

Tym samym pokazano, iż dyskretne N-punktowe przekształcenia kosinusowe i sinusowe drugiego i czwartego rodzaju umożliwiają przybliżone obliczanie widm całkowych przekształceń kosinusowych i sinusowych Fouriera w punktach wynikających z odpowiednich reguł dyskretyzacji, tj. $\omega_k^{II} = k\Delta\omega'$ lub $\omega_k^{IV} = (k+\frac{1}{2})\Delta\omega'$ dla parametru $k = 0, 1, ..., N-1$. Przy czym przekształcenia te operują na dyskretnym zbiorze próbek transformowanego sygnału $x(t)$ postaci $\left\{x((n+\frac{1}{2})\Delta t) : n = 0, 1, ..., N-1\right\}$.

Odtworzenie sygnału $x(t)$ w punktach $t'_n = (n+\frac{1}{2})\Delta t$, na podstawie zbiorów współczynników przekształceń: DCT-II, DCT-IV oraz DST-II i DST-IV, jest możliwe przy użyciu dyskretnych przekształceń odwrotnych, które w zależności od typu przekształcenia prostego definiuje się jako [4]

$$x(n) = IDCTII_N\left\{X_N^{CII}(k)\right\} \triangleq \sum_{k=0}^{N-1} X_N^{CII}(k)cos\left(\frac{\pi}{N}(n+\frac{1}{2})k\right)$$

dla przekształcenia DCT-II,

$$x(n) = IDCTIV_N\left\{X_N^{CIV}(k)\right\} \triangleq \sum_{k=0}^{N-1} X_N^{CIV}(k)cos\left(\frac{\pi}{N}(n+\frac{1}{2})(k+\frac{1}{2})\right)$$

dla przekształcenia DCT-IV,

$$x(n) = IDSTII_N\left\{X_N^{SII}(k)\right\} \triangleq \sum_{k=0}^{N-1} X_N^{SII}(k)sin\left(\frac{\pi}{N}(n+\frac{1}{2})(k+1)\right)$$

dla przekształcenia DST-II, oraz

$$x(n) = IDSTIV_N\left\{X_N^{SIV}(k)\right\} \triangleq \sum_{k=0}^{N-1} X_N^{SIV}(k)sin\left(\frac{\pi}{N}(n+\frac{1}{2})(k+\frac{1}{2})\right)$$

dla przekształcenia DST-IV, gdzie $n = 0, 1, ..., N-1$.

W dalszej części książki dla uproszczenia zapisu przyjmowane będzie następujące założenie $T = 1$. Na mocy tw. o zmianie skali w dziedzinie czasu [14, 136] można każdy przedział $[0, T]$ sprowadzić do przypadku $[0, 1]$. Zatem powyższe założenie nie powoduje utraty ogólności rozważań. Ponadto widmo sygnału $X_p(k)$ ($X_p^C(k), X_p^S(k)$) często utożsamiane będzie z $X_N(k)$ ($X_N^{CII}(k)$ lub $X_N^{CIV}(k), X_N^{SII}(k)$ lub $X_N^{SIV}(k)$), a próbki sygnału $x(t)$ pobierane w punktach t_n (t'_n), oznaczane będą jako $x(n)$. Wskazane założenia wpływać będą na uproszczenie zapisu i zależeć od rozpatrywanego przekształcenia.

2.4. Związki symetrii dla dyskretnych przekształceń kosinusowych i sinusowych

W czwartym rozdziale monografii podczas syntezy szybkich algorytmów obliczania dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju wykorzystane zostaną własności symetrii w grupie rozpatrywanych przekształceń, definiowane względem pewnych punktów w dziedzinie częstotliwości. Stąd w bieżącej sekcji zestawiono wspomniane własności wraz z ich dowodami.

Własność 2.4.1 *(Związek pomiędzy przekształceniami DST-II a DCT-II)*
Dla dyskretnego N-punktowego przekształcenia sinusowego drugiego rodzaju w postaci $X_N^{SII}(k) = DSTII_N\{x(n)\}$, operującego na dyskretnym zbiorze $\{x(n)\}$, prawdziwa jest następująca równość $X_N^{SII}(N-k-1) = DCTII_N\{(-1)^n x(n)\}$ dla $k = 0, 1, ..., N-1$.

Dowód. Powyższą własność dowodzi się w sposób następujący

$$\begin{aligned} X_N^{SII}(N-k-1) &= \frac{p_k}{N}\left(\sum_{n=0}^{N-1} x(n) sin(\frac{\pi}{N}(N-k-1+1)(n+\frac{1}{2}))\right) = \\ &= \frac{p_k}{N}\left(\sum_{n=0}^{N-1} x(n) sin(\pi(n+\frac{1}{2}) - \frac{\pi}{N}k(n+\frac{1}{2}))\right) = \\ &= \frac{p_k}{N}\left(\sum_{n=0}^{N-1} (-1)^n x(n) cos(\frac{\pi}{N}k(n+\frac{1}{2}))\right) \end{aligned}$$

dla $k = 0, 1, ..., N-1$. Z prawej strony otrzymanej zależności widnieje dyskretne przekształcenie kosinusowe drugiego rodzaju, które operuje na ciągu próbek $x(n)$ przeskalowanych przez czynnik $(-1)^n$. ∎

Własność 2.4.2 *(Symetria przekształcenia DCT-II względem punktu $k = 2N$)*
Jeżeli mamy dane dyskretne N-punktowe przekształcenie kosinusowe drugiego rodzaju $X_N^{CII}(k) = DCTII_N\{x(n)\}$, operujące na zbiorze $\{x(n)\}$, to dla $k = 0, 1, ..., N-1$ zachodzi równość $X_N^{CII}(2N-k) = -X_N^{CII}(k)$.

Dowód. Dowód powyższej własności przyjmuje następującą postać

$$\begin{aligned} X_N^{CII}(2N-k) &= \frac{p_k}{N}\left(\sum_{n=0}^{N-1} x(n) cos(\frac{\pi}{N}(2N-k)(n+\frac{1}{2}))\right) = \\ &= \frac{p_k}{N}\left(\sum_{n=0}^{N-1} x(n) cos(2\pi(n+\frac{1}{2}) - \frac{\pi}{N}k(n+\frac{1}{2}))\right) = \\ &= -\frac{p_k}{N}\left(\sum_{n=0}^{N-1} x(n) cos(\frac{\pi}{N}k(n+\frac{1}{2}))\right) \end{aligned}$$

dla $k = 0, 1, ..., N-1$. Po prawej stronie powyższego wyrażenia otrzymujemy dyskretne przekształcenie kosinusowe drugiego rodzaju dla ciągu $x(n)$, które brane jest ze znakiem minus. ∎

2.5. Przypadek przekształceń dwuwymiarowych

W praktyce cyfrowego przetwarzania danych, obok sygnałów jednowymiarowych, równie często przekształceniom podlegają sygnały dwuwymiarowe, tzn. sygnały określone na płaszczyźnie. Jako przykłady sygnałów dwuwymiarowych posłużyć mogą sekwencje sygnałów jednowymiarowych obserwowanych w pewnych przedziałach czasu, a także obrazy cyfrowe, w tym obrazy świata naturalnego rejestrowane w zakresie widma światła widzialnego oraz w bliskiej i dalekiej podczerwieni, obrazy medyczne, obrazy generowane w sposób sztuczny, etc. Sygnały takie opisuje się matematycznie jako funkcje dwóch zmiennych.

Niech zatem $x(t,s)$ oznacza funkcję dwóch zmiennych $t,s \in \mathcal{R}$, która przyjmuje wartości różne od zera jedynie na przedziale $[0,T_1] \times [0,T_2]$ dla pewnych $T_1, T_2 \in \mathcal{R}$. Wówczas zgodnie z pracą [14] dwuwymiarowe przekształcenie Fouriera w postaci całkowej definiujemy jako

$$X^{2D}(\omega,\Omega) = \mathcal{F}_{2D}\left\{x(t,s)\right\} \triangleq \int_0^{T_1}\int_0^{T_2} x(t,s)e^{-i(\omega t+\Omega s)}dtds, \tag{2.15}$$

gdzie parametry $\omega, \Omega \in \mathcal{R}$ oznaczają pulsacje. Natomiast odwrotne dwuwymiarowe przekształcenie Fouriera przyjmuje postać wyrażenia [14]

$$x(t,s) = \mathcal{F}_{2D}^{-1}\left\{X(\omega,\Omega)\right\} \triangleq \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X(\omega,\Omega)e^{i(\omega t+\Omega s)}d\omega d\Omega. \tag{2.16}$$

Zakładając podobnie jak w przypadku jednowymiarowym, że funkcja $x(t,s)$ jest przyczynową funkcją czasu, tzn. jest tożsamościowo równa zeru na płaszczyźnie $\{(t,s)\}$ dla $t<0$ lub $s<0$, możemy podać pary dwuwymiarowych transformat: kosinusowej i sinusowej Fouriera w postaci całkowej. W rozważanym przypadku przyjmą one odpowiednio postaci całek

$$X^{C2D}(\omega,\Omega) = \mathcal{F}_{C2D}\left\{x(t,s)\right\} \triangleq \int_0^{T_1}\int_0^{T_2} x(t,s)cos(\omega t)cos(\Omega s)dtds, \tag{2.17a}$$

$$x(t,s) = \mathcal{F}_{C2D}^{-1}\left\{X^{C2D}(\omega,\Omega)\right\} \triangleq \frac{4}{\pi^2}\int_0^{\infty}\int_0^{\infty} X^{C2D}(\omega,\Omega)cos(\omega t)cos(\Omega s)d\omega d\Omega \tag{2.17b}$$

dla dwuwymiarowego kosinusowego przekształcenia Fouriera oraz

$$X^{S2D}(\omega,\Omega) = \mathcal{F}_{S2D}\left\{x(t,s)\right\} \triangleq \int_0^{T_1}\int_0^{T_2} x(t,s)sin(\omega t)sin(\Omega s)dtds, \tag{2.18a}$$

$$x(t,s) = \mathcal{F}_{S2D}^{-1}\left\{X^{S2D}(\omega,\Omega)\right\} \triangleq \frac{4}{\pi^2}\int_0^{\infty}\int_0^{\infty} X^{S2D}(\omega,\Omega)sin(\omega t)sin(\Omega s)d\omega d\Omega \tag{2.18b}$$

w przypadku dwuwymiarowego sinusowego przekształcenia Fouriera.

Następnie dla wykazania związków pomiędzy wymienionymi całkowymi przekształceniami trygonometrycznymi, a dwuwymiarowym dyskretnym przekształceniem Fouriera, oraz dwuwymiarowymi dyskretnymi przekształceniami kosinusowymi i sinusowymi drugiego i czwartego rodzaju, postępuje się w sposób analogiczny do przypadku

jednowymiarowego, tzn. należy wprowadzić do wzorów (2.15) - (2.18b) odpowiednio dobrane dyskretyzacje w dziedzinie czasu i w dziedzinie częstotliwości.

Niech zatem na przedział $[0, T_1] \times [0, T_2]$, na którym funkcja $x(t, s)$ może przyjmować wartości niezerowe, przypada N na M próbek (odpowiednio względem zmiennych t i s), gdzie N i M to liczby naturalne. Dobieramy wówczas kroki dyskretyzacji w dziedzinie czasu w postaci stałych $\Delta t = T_1/N$ oraz $\Delta s = T_2/M$. Jako pierwszy rozpatrzony zostanie przypadek dwuwymiarowego przekształcenia Fouriera definiowanego jako wyrażenie postaci (2.15).

W tym celu zakładamy dobór próbek w punktach (t_n, s_m) określonych przez wyrażenia $t_n = n\Delta t$ i $s_m = m\Delta s$ dla parametrów $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$. Natomiast jako reguły dyskretyzacji w dziedzinie częstotliwości przyjmujemy wyrażenia $\omega_k = k\Delta\omega$ oraz $\Omega_l = \Delta\Omega$ dla $k = 0, \pm 1, ..., \pm\infty$ i $l = 0, \pm 1, ..., \pm\infty$, z krokami dyskretyzacji postaci $\Delta\omega = 2\pi/T_1$ oraz $\Delta\Omega = 2\pi/T_2$. Po podstawieniu tak zdefiniowanych reguł do wyrażenia (2.15) otrzymujemy wzór pozwalający na przybliżone obliczenie przekształcenia Fouriera, na podstawie zbioru dyskretnych próbek $\{x(t_n, s_m) : n = 0, 1, ..., N-1, m = 0, 1, ..., M-1\}$ wejściowego sygnału $x(t, s)$

$$X^{2D}(k\Delta\omega, l\Delta\Omega) \approx \frac{T_1 T_2}{NM} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(n\Delta t, m\Delta s) e^{-i\frac{2\pi}{N}kn} e^{-i\frac{2\pi}{M}lm} =$$

$$= T_1 T_2 DFT2D_{N \cdot M} \{x(n\Delta t, m\Delta s)\}$$

dla $k = 0, \pm 1, ..., \pm\frac{N}{2}$ i $m = 0, \pm 1, ..., \pm\frac{M}{2}$. W przypadku ogólnym powyższe wyrażenie pozwala jedynie na przybliżone obliczanie wartości $X^{2D}(k\Delta\omega, l\Delta\Omega)$ całki dwuwymiarowego przekształcenia Fouriera, co znów spowodowane jest występowaniem zjawiska aliasingu. Widoczny w powyższym wzorze symbol $DFT2D_{N \cdot M} \{x(n, m)\}$ oznacza dwuwymiarowe N na M-punktowe dyskretne przekształcenia Fouriera (DFT2D), które definiowane jest zgodnie z pracą [154] jako

$$X^{2D}_{N \cdot M}(k, l) = DFT2D_{N \cdot M} \{x(n, m)\} \triangleq \frac{1}{NM} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(n, m) e^{-i\frac{2\pi}{N}kn} e^{-i\frac{2\pi}{M}lm}. \quad (2.19)$$

Z porównania powyższych zależności wynika, iż przekształcenie DFT2D (z dokładnością do stałej $T_1 T_2$) umożliwia przybliżone obliczanie widma $X^{2D}(\omega, \Omega)$ sygnału $x(t, s)$ dla dyskretnych wartości pulsacji (ω_k, Ω_l), które realizowane jest na podstawie N na M-elementowego zbioru dyskretnych próbek sygnału $\{x(t_n, s_m)\}$.

Odtworzenie sygnału wejściowego na podstawie zbioru współczynników $X^{2D}_{N \cdot M}(k, l)$ w punktach (t_n, s_m), jest możliwe przy użyciu odwrotnego dwuwymiarowego dyskretnego przekształcenia Fouriera (IDFT2D), którego postać definiuje poniższa zależność

$$x(n, m) = IDFT2D_{N \cdot M} \left\{ X^{2D}_{N \cdot M}(k, l) \right\} \triangleq \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} X^{2D}_{N \cdot M}(k, l) e^{i\frac{2\pi}{N}kn} e^{i\frac{2\pi}{M}lm}$$

dla $n = 0, 1, ..., N-1$ oraz $m = 0, 1, ..., M-1$.

Z kolei stosując odmienny schemat doboru próbek sygnału w czasie oparty o reguły $t_n = (n + 1/2)\Delta t$ i $s_m = (m + 1/2)\Delta s$, a także wprowadzając reguły dyskretyzacji w dziedzinie częstotliwości definiowane odpowiednio jako

$$\omega^{II}_k = k\Delta\omega', \ \Omega^{II}_l = l\Delta\Omega',$$

$$\omega^{IV}_k = (k + \tfrac{1}{2})\Delta\omega', \ \Omega^{IV}_l = (l + \tfrac{1}{2})\Delta\Omega'$$

dla $k = 0, 1, ..., \infty$ i $l = 0, 1, ..., \infty$, gdzie $\Delta\omega' = \pi/T_1$ oraz $\Delta\Omega' = \pi/T_2$ to kroki dyskretyzacji w dziedzinie częstotliwości, można wskazać analogiczne związki pomiędzy całkowymi przekształceniami kosinusowymi i sinusowymi, a dyskretnymi przekształceniami kosinusowymi i sinusowymi drugiego (dyskretyzacja wg. $\omega_k^{II}, \Omega_l^{II}$) rodzaju

$$X^{C2D}(k\Delta\omega', l\Delta\Omega') \approx \frac{T_1T_2}{p_kp_l} DCT2DII_{N\cdot M}\left\{x((n+\frac{1}{2})\Delta t, (m+\frac{1}{2})\Delta s)\right\},$$

$$X^{S2D}((k+1)\Delta\omega', (l+1)\Delta\Omega') \approx \frac{T_1T_2}{p_kp_l} DST2DII_{N\cdot M}\left\{x((n+\frac{1}{2})\Delta t, (m+\frac{1}{2})\Delta s)\right\}$$

dla $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$ oraz czwartego (dyskretyzacja wg. $\omega_k^{IV}, \Omega_l^{IV}$) rodzaju

$$X^{C2D}((k+\tfrac{1}{2})\Delta\omega', (l+\tfrac{1}{2})\Delta\Omega') \approx T_1T_2 DCT2DIV_{N\cdot M}\left\{x((n+\tfrac{1}{2})\Delta t, (m+\tfrac{1}{2})\Delta s)\right\},$$

$$X^{S2D}((k+\tfrac{1}{2})\Delta\omega', (l+\tfrac{1}{2})\Delta\Omega') \approx T_1T_2 DST2DIV_{N\cdot M}\left\{x((n+\tfrac{1}{2})\Delta t, (m+\tfrac{1}{2})\Delta s)\right\}$$

dla $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$. Symbol $DCT2DII_{N\cdot M}$ oznacza dyskretne dwuwymiarowe N na M-punktowe przekształcenie kosinusowe drugiego rodzaju, definiowane jako (patrz [110])

$$X_{N\cdot M}^{C2DII}(k,l) = DCT2DII_{N\cdot M}\left\{x(n,m)\right\}, \tag{2.20}$$
$$X_{N\cdot M}^{C2DII}(k,l) \triangleq \frac{p_kp_l}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)cos(\frac{\pi}{N}k(n+\frac{1}{2}))cos(\frac{\pi}{M}l(m+\frac{1}{2}))$$

dla $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$, natomiast $DST2DII_{N\cdot M}$, $DCT2DIV_{N\cdot M}$ oraz $DST2DIV_{N\cdot M}$ to odpowiednio dyskretne dwuwymiarowe N na M-punktowe przekształcenia sinusowe drugiego rodzaju oraz kosinusowe i sinusowe rodzaju czwartego, definiowane zgodnie z pracą [110] jako

$$X_{N\cdot M}^{S2DII}(k,l) = DST2DII_{N\cdot M}\left\{x(n,m)\right\}, \tag{2.21}$$
$$X_{N\cdot M}^{S2DII}(k,l) \triangleq \frac{p_kp_l}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)sin(\frac{\pi}{N}(k+1)(n+\frac{1}{2}))sin(\frac{\pi}{M}(l+1)(m+\frac{1}{2}))$$

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

$$X_{N\cdot M}^{C2DIV}(k,l) = DCT2DIV_{N\cdot M}\left\{x(n,m)\right\}, \tag{2.22}$$
$$X_{N\cdot M}^{C2DIV}(k,l) \triangleq \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)cos(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2}))cos(\frac{\pi}{M}(l+\frac{1}{2})(m+\frac{1}{2}))$$

dla $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$ oraz

$$X_{N\cdot M}^{S2DIV}(k,l) = DST2DIV_{N\cdot M}\left\{x(n,m)\right\}, \tag{2.23}$$
$$X_{N\cdot M}^{S2DIV}(k,l) \triangleq \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)sin(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2}))sin(\frac{\pi}{M}(l+\frac{1}{2})(m+\frac{1}{2}))$$

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

Odtworzenie sygnału wejściowego w punktach (t'_n, t'_m) jest możliwe na podstawie zbiorów współczynników przekształceń: DCT2D-II, DCT2D-IV oraz DST2D-II i DST2D-IV, przy użyciu przekształceń odwrotnych, które zgodnie z pracą [110] definiują poniższe zależności

$$x(n,m) = IDCT2DII_{N\cdot M}\left\{X_{N\cdot M}^{C2DII}(k,l)\right\},$$

$$x(n,m) \triangleq \sum_{k=0}^{N-1}\sum_{l=0}^{M-1} X_{N\cdot M}^{C2DII}(k,l) cos\left(\frac{\pi}{N}(n+\frac{1}{2})k\right) cos\left(\frac{\pi}{M}(m+\frac{1}{2})l\right)$$

dla przekształcenia DCT2D-II,

$$x(n,m) = IDCT2DIV_{N\cdot M}\left\{X_{N\cdot M}^{C2DIV}(k,l)\right\},$$

$$x(n,m) \triangleq \sum_{k=0}^{N-1}\sum_{l=0}^{M-1} X_{N\cdot M}^{C2DII}(k,l) cos\left(\frac{\pi}{N}(n+\frac{1}{2})(k+\frac{1}{2})\right) cos\left(\frac{\pi}{M}(m+\frac{1}{2})(l+\frac{1}{2})\right)$$

dla przekształcenia DCT2D-IV, oraz

$$x(n,m) = IDST2DII_{N\cdot M}\left\{X_{N\cdot M}^{S2DII}(k,l)\right\},$$

$$x(n,m) \triangleq \sum_{k=0}^{N-1}\sum_{l=0}^{M-1} X_{N\cdot M}^{S2DII}(k,l) sin\left(\frac{\pi}{N}(n+\frac{1}{2})(k+1)\right) sin\left(\frac{\pi}{M}(m+\frac{1}{2})(l+1)\right)$$

dla przekształcenia DST2D-II i

$$x(n,m) = IDST2DIV_{N\cdot M}\left\{X_{N\cdot M}^{S2DIV}(k,l)\right\},$$

$$x(n,m) \triangleq \sum_{k=0}^{N-1}\sum_{l=0}^{M-1} X_{N\cdot M}^{S2DII}(k,l) sin\left(\frac{\pi}{N}(n+\frac{1}{2})(k+\frac{1}{2})\right) sin\left(\frac{\pi}{M}(m+\frac{1}{2})(l+\frac{1}{2})\right)$$

dla przekształcenia DST2D-IV, gdzie $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$.

Proces syntezy szybkich algorytmów obliczania dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju, przeprowadzony w rozdziale 4, oparty został na pewnych własnościach symetrii w grupie omawianych dyskretnych przekształceń dwuwymiarowych. Dlatego też w dalszej części niniejszego rozdziału zestawiono wspomniane własności w postaci twierdzeń.

Własność 2.5.1 *(Antysymetria przekształcenia DCT2D-II względem punktu $(2N, l)$) Jeżeli dane jest dwuwymiarowe dyskretne N na M-punktowe przekształcenie kosinusowe drugiego rodzaju $X_{N\cdot M}^{C2DII}(k,l) = DCT2DII_{N\cdot M}\{x(n,m)\}$, określone na zbiorze elementów $\{x(n,m)\}$, to wówczas $X_{N\cdot M}^{C2DII}(2N-k,l) = -X_{N\cdot M}^{C2DII}(k,l)$ dla każdej pary (k,l), gdzie $k = 0, 1, ..., N-1$ oraz $l = 0, 1, ..., M-1$.*

Dowód. Prawdziwość powyższej własności dowodzą następujące przekształcenia

$$X_{N\cdot M}^{C2DII}(2N-k,l) = \frac{p_k p_l}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m) cos(\frac{\pi}{N}(2N-k)(n+\frac{1}{2})) cos(\frac{\pi}{M}l(m+\frac{1}{2})) =$$

$$= \frac{p_k p_l}{NM} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(n,m) cos(2\pi(n+\frac{1}{2}) - \frac{\pi}{N}k(n+\frac{1}{2}))cos(\frac{\pi}{M}l(m+\frac{1}{2})) =$$

$$= \frac{p_k p_l}{NM} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(n,m) cos(\pi - \frac{\pi}{N}k(n+\frac{1}{2}))cos(\frac{\pi}{M}l(m+\frac{1}{2}))$$

$$= -X_{N\cdot M}^{C2DII}(k,l)$$

dla $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$. ∎

W sposób analogiczny można wykazać prawdziwość kolejnej własności.

Własność 2.5.2 *(Antysymetria przekształcenia DCT2D-II względem punktu $(k, 2M)$) Mając dane dwuwymiarowe dyskretne N na M-punktowe przekształcenie kosinusowe drugiego rodzaju $X_{N\cdot M}^{C2DII}(k,l) = DCT2DII_{N\cdot M}\{x(n,m)\}$, określone na zbiorze elementów $\{x(n,m)\}$, to wówczas $X_{N\cdot M}^{C2DII}(k, 2M-l) = -X_{N\cdot M}^{C2DII}(k,l)$ dla parametrów $k = 0, 1, ..., N-1$ oraz $l = 0, 1, ..., M-1$.*

W świetle własności (2.5.1) i (2.5.2) można sformułować poniższą własność symetrii dla przekształcenia DCT2D-II. Otóż:

Własność 2.5.3 *(Symetria przekształcenia DCT2D-II względem punktu $(2N, 2M)$) Dla dwuwymiarowego dyskretnego N na M-punktowego przekształcenia kosinusowego drugiego rodzaju $X_{N\cdot M}^{C2DII}(k,l) = DCT2DII_{N\cdot M}\{x(n,m)\}$, określonego na zbiorze elementów $\{x(n,m)\}$, zachodzi związek $X_{N\cdot M}^{C2DII}(2M-k, 2M-l) = X_{N\cdot M}^{C2DII}(k,l)$ dla $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$.*

Własność 2.5.4 *(Symetria przekształcenia DCT2D-II względem punktu $(0, l)$) Dwuwymiarowe dyskretne N na M-punktowe przekształcenie kosinusowe drugiego rodzaju $X_{N\cdot M}^{C2DII}(k,l) = DCT2DII_{N\cdot M}\{x(n,m)\}$, określone na zbiorze $\{x(n,m)\}$, posiada własność symetrii postaci $X_{N\cdot M}^{C2DII}(-k,l) = X_{N\cdot M}^{C2DII}(k,l)$ dla $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$.*

Analogiczne własności można podać dla punktów $(k, 0)$ oraz $(0, 0)$.

Własność 2.5.5 *(Symetria przekształcenia DCT2D-II względem punktu $(k, 0)$) Dla dwuwymiarowego dyskretnego N na M-punktowego przekształcenia kosinusowego drugiego rodzaju $X_{N\cdot M}^{C2DII}(k,l) = DCT2DII_{N\cdot M}\{x(n,m)\}$, określonego na zbiorze elementów $\{x(n,m)\}$, prawdziwa jest własność postaci $X_{N\cdot M}^{C2DII}(k,-l) = X_{N\cdot M}^{C2DII}(k,l)$ dla parametrów $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$.*

Własność 2.5.6 *(Symetria przekształcenia DCT2D-II względem punktu $(0, 0)$) Dwuwymiarowe dyskretne N na M-punktowe przekształcenie kosinusowe drugiego rodzaju $X_{N\cdot M}^{C2DII}(k,l) = DCT2DII_{N\cdot M}\{x(n,m)\}$, określone na zbiorze $\{x(n,m)\}$, charakteryzuje własność symetrii $X_{N\cdot M}^{C2DII}(-k,-l) = X_{N\cdot M}^{C2DII}(k,l)$ dla $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$.*

Dowody własności (2.5.4) - (2.5.6) wynikają bezpośrednio z własności symetrii funkcji kosinus oraz z własności separowalności dwuwymiarowych przekształceń kosinusowych (patrz [154]).

2.6. Podsumowanie i wnioski

Na wstępie niniejszego rozdziału przedstawiono definicje jednowymiarowych przekształceń trygonometrycznych Fouriera oraz kosinusowego i sinusowego przekształcenia Fouriera w postaci całkowej, a także krótko omówiono warunki istnienia oraz wzajemnej jednoznaczności wspomnianych przekształceń. Następnie poprzez wprowadzenie odpowiednich reguł doboru próbek dla sygnału wejściowego w dziedzinie czasu, oraz dyskretyzacji w dziedzinie częstotliwości, wskazano związki pomiędzy rozważanymi przekształceniami całkowymi, a dyskretnymi przekształceniami Fouriera oraz kosinusowymi i sinusowymi przekształceniami drugiego i czwartego rodzaju.

Końcową część rozdziału poświęcono przypadkom przekształceń dwuwymiarowych. Tutaj, podobnie jak dla przypadku jednowymiarowego, podano związki występujące pomiędzy całkowymi przekształceniami Fouriera, a dyskretnymi dwuwymiarowymi przekształceniami Fouriera oraz kosinusowymi i sinusowymi przekształceniami obu rozpatrywanych rodzajów. Ponadto dla obu przypadków przekształceń jedno- i dwuwymiarowych zamieszczono odpowiednie własności symetrii w grupie dyskretnych przekształceń kosinusowych i sinusowych, które wykorzystane zostaną w rozdziale czwartym do syntezy szybkich algorytmów obliczania wspomnianych przekształceń dyskretnych.

Z rozważań zawartych w niniejszym rozdziale bezpośrednio wynikają następujące wnioski:

- W przypadku ogólnym, dla sygnału spróbkowanego w czasie, możliwe jest jedynie przybliżone obliczanie widm całkowych przekształceń Fouriera oraz kosinusowego i sinusowego przekształcenia Fouriera, które w zależności od doboru reguł dyskretyzacji w dziedzinie czasu i w dziedzinie częstotliwości, oblicza się przy pomocy dyskretnych przekształceń Fouriera lub kosinusowego i sinusowego przekształcenia drugiego i czwartego rodzaju.

- Błąd obliczania przekształcenia całkowego dla dyskretnych wartości pulsacji powstaje w wyniku powielania widma (tzw. zjawisko aliasingu) sygnału z całkowitymi wielokrotnościami częstotliwości $f_p = 1/\Delta t$ próbkowania w dziedzinie czasu, a wartość tego błędu zależy wprost proporcjonalnie od kroku dyskretyzacji Δt. Zatem zmniejszając krok dyskretyzacji można zmniejszyć błąd obliczania widma, kosztem większej liczby operacji arytmetycznych. Wynika to z faktu, iż zmniejszaniu kroku dyskretyzacji towarzyszy wzrost liczby próbek sygnału. Stąd dobór właściwego kroku dyskretyzacji należy do kluczowych problemów cyfrowego przetwarzania i analizy danych.

- Analogiczne wnioski można sformułować dla dwuwymiarowych przekształceń trygonometrycznych.

Rozpatrywane związki pomiędzy przekształceniami całkowymi i dyskretnymi można w podobny sposób przenieść na większą niż dwa liczbę wymiarów dziedziny określoności sygnału wejściowego.

3

Numeryczne obliczanie przekształceń trygonometrycznych

Praktyczne rozwiązania wielu problemów z zakresu inżynierii, matematyki stosowanej, fizyki, diagnostyki technicznej i medycznej, a także innych zagadnień, które wymagają przedstawienia sygnałów w dziedzinach przekształceń trygonometrycznych, często przeprowadza się z wykorzystaniem komputerów oraz cyfrowych procesorów sygnałowych. W takich przypadkach, ze względu na dyskretny charakter danych, które pozyskiwane są w postaci skończonych zbiorów dyskretnych próbek przetwarzanych sygnałów, nie jest możliwe zastosowanie analitycznego aparatu matematycznego. Wówczas do rozwiązania postawionych problemów wykorzystuje się przybliżone rozwiązania oparte o analizę numeryczną.

W poprzednim rozdziale wykazano, iż dyskretne przekształcenia trygonometryczne w postaci przekształcenia Fouriera oraz kosinusowych i sinusowych przekształceń drugiego i czwartego rodzaju, w przypadku ogólnym, pozwalają jedynie na przybliżone obliczanie całkowych przekształceń Fouriera w punktach, wynikających z przyjętych reguł dyskretyzacji w dziedzinie częstotliwości. Niniejszy rozdział poświęcono zagadnieniom numerycznego obliczania jedno- i dwuwymiarowych całkowych przekształceń trygonometrycznych, przy czym wykazano, iż wspomniane dyskretne przekształcenia trygonometryczne stanowią kwadratury numerycznego obliczania przekształceń Fouriera w postaci całkowej.

W sekcji 3.1 zamieszczono ogólną dyskusję problematyki numerycznego obliczania całek z wykorzystaniem kwadratur numerycznych, oraz przedstawiono definicje złożonych kwadratur rzędu pierwszego, wraz z wzorami oceny błędu przybliżonego, numerycznego obliczania wartości całek. Następnie wykazano, iż dyskretne przekształcenia trygonometryczne, w postaci przekształcenia Fouriera oraz kosinusowych i sinusowych przekształceń drugiego i czwartego rodzaju, są złożonymi kwadraturami pierwszego rzędu, tzn. stanowią złożone wzory trapezów dla przybliżonego numerycznego obliczania przekształceń trygonometrycznych w postaci całkowej. Dzięki tej własności możliwe jest przeniesienie na grunt obliczania przekształceń trygonometrycznych pewnych rozwiązań z zakresu całkowania numerycznego, w szczególności algorytmów adaptacyjnego obliczania całek. Stąd w sekcji 3.3 zamieszczono propozycje algorytmów adaptacyjnych

dla rozważanych dyskretnych przekształceń trygonometrycznych, które umożliwiają automatyczny dobór liczby próbek sygnału zgodnie z kryterium *dokładności - czasu realizacji obliczeń* całkowych przekształceń trygonometrycznych.

Przeprowadzone w sekcji 3.1 rozważania dotyczące przekształceń jednowymiarowych, zostały rozszerzone w sekcji 3.2 na przypadki przekształceń dwuwymiarowych. Ostatnią część niniejszego rozdziału (patrz sekcja 3.4) wypełnia opis algorytmów adaptacyjnych dla przypadku dyskretnych dwuwymiarowych przekształceń trygonometrycznych, które rozważa się jako kubatury całkowania numerycznego dla dwuwymiarowych przekształceń trygonometrycznych w postaci całkowej.

3.1. Dyskretne przekształcenia trygonometryczne jako kwadratury numeryczne

U podstaw numerycznego obliczania całek postaci $\int_a^b f(t)dt$ leży założenie o aproksymacji funkcji podcałkowej $f(t)$ prostszą funkcją $\hat{f}(t)$, dla której analityczna postać wyrażenia $\int_a^b \hat{f}(t)dt$ jest znana. Wówczas w przypadku ogólnym zachodzi następujący związek

$$I(f) = \int_a^b f(t)dt \approx \int_a^b \hat{f}(t)dt, \tag{3.1}$$

gdzie całka $\int_a^b \hat{f}(t)dt$ pełni rolę przybliżenia wartości całki z funkcji $f(t)$. Przybliżenie to nosi miano *kwadratury całkowania numerycznego* (patrz [7, 32, 40]).

Najprostszym sposobem rozwiązania zadania (3.1) jest aproksymacja funkcji podcałkowej wielomianem interpolacyjnym $\hat{f(t)}$, np. wielomianem Lagrange'a, opartym na skończonym zbiorze wartości funkcji $f(t)$, zwanych *współczynnikami węzłowymi*. Współczynniki te pobiera się w dyskretnych punktach $\{t_0, t_1, ..., t_{N-1}\}$ zwanych *punktami węzłowymi kwadratury*, które wybiera się w taki sposób, ażeby spełniona była następująca nierówność: $a \leqslant t_0 < t_1 < ... < t_{N-1} \leqslant b$, dla pewnego N będącego dodatnią liczbą całkowitą. Wówczas zgodnie z twierdzeniem *Weierstrassa*† (patrz [112]) dobierając właściwy wielomian interpolacyjny dostatecznie wysokiego stopnia r, jesteśmy w stanie dowolnie zmniejszyć błąd aproksymacji funkcji $f(t)$ tym wielomianem, jednocześnie zmniejszając błąd przybliżenia wartości całki $I(f)$ przez kwadraturę $\int_a^b \hat{f}(t)dt$. Kwadraturę dającą dokładną wartość $I(f)$ dla funkcji $f(t)$, będącej wielomianem co najwyżej stopnia r, nazywamy *kwadraturą rzędu r*.

W praktyce rzadko stosuje się podejście bazujące na wielomianach wysokich stopni, przez wzgląd na możliwość wystąpienia tzw. zjawiska Rungego (patrz [7, 32]). Zjawisko to objawia się wzrostem błędu aproksymacji podczas zwiększania stopnia r wielomianu interpolacyjnego. Zwiększanie błędu aproksymacji spowodowane jest wzrostem wartości $(r+1)$-szej pochodnej funkcji $f(t)$, od której błąd aproksymacji zależy w sposób wprost proporcjonalny. Chociaż zjawisko Rungego będzie występować jedynie dla pewnej klasy funkcji, których pochodne wyższych rzędów zachowują się w powyżej wspomniany sposób, to w zadaniach, w których nie dysponuje się dostateczną wiedzą na temat zachowania pochodnych funkcji podcałkowej, preferuje się rozwiązania oparte

† *Twierdzenie Weierstrassa.*
Każda funkcja ciągła określona na pewnym przedziale $[a, b]$ może być aproksymowana w tym przedziale jednostajnie i z dowolną dokładnością wielomianem dostatecznie wysokiego stopnia.

o wielomiany niższych stopni, na przykład stopni $r = 1, 2$ lub 3. W takich przypadkach wielomian interpolacyjny $\hat{f}(t)$ przyjmuje postać sumy wielomianów stopni niższych, a błąd aproksymacji funkcji podcałkowej $f(t)$ można redukować poprzez zagęszczanie punktów węzłowych $\{t_0, t_1, ..., t_{N-1}\}$, tzn. zmniejszając odległości $h_n = (t_{n+1} - t_n)$ pomiędzy poszczególnymi punktami t_n, które przypadają na przedział całkowania $[a, b]$. Operacja zagęszczania węzłów wymaga zwiększenia ich całkowitej liczby N. Kwadratury numeryczne budowane na bazie tak skonstruowanych wielomianów noszą nazwę *kwadratur złożonych.*

Mając na uwadze tematykę niniejszej monografii, która dotyczy dyskretnych przekształceń trygonometrycznych, w dalszej części niniejszej sekcji skupiono się wyłącznie na złożonych kwadraturach rzędu pierwszego. Na wstępie przedstawione zostaną definicje równomiernych kwadratur złożonych trapezów i prostokątów, wraz z wzorami określającymi błędy przybliżania wartości całek poprzez te kwadratury. Następnie wykazane zostaną podobieństwa pomiędzy rozważanymi kwadraturami, a jednowymiarowymi dyskretnymi przekształceniami trygonometrycznymi.

Przyjmujemy, że funkcja podcałkowa $f(t)$ jest ciągła i określona na przedziale $[a, b]$, może przyjmować wyłącznie wartości rzeczywiste i posiada co najmniej K ciągłych i ograniczonych pochodnych. Zakładamy dodatkowo, że K jest liczbą parzystą. Niech $\{t_n : n = 0, 1, ..., N\}$ oznacza zbiór $N+1$ punktów węzłowych przypadających na przedział $[a, b]$ i dobranych w taki sposób, że $a = t_0 < t_1 < ... < t_{N-2} < t_{N-1} < t_N = b$. Dla dalszych rozważań wybieramy dwa sąsiednie punkty węzłowe t_n oraz t_{n+1} dla $0 \leqslant n < N$. Wówczas wielomian interpolacyjny Lagrange'a pierwszego stopnia $L_n(f)$, oparty na tak dobranych punktach węzłowych, przyjmie postać wyrażenia [7, 32, 40]

$$L_n(f) = f(t_n)\frac{(t - t_{n+1})}{(t_n - t_{n+1})} + f(t_{n+1})\frac{(t - t_n)}{(t_{n+1} - t_n)}.$$

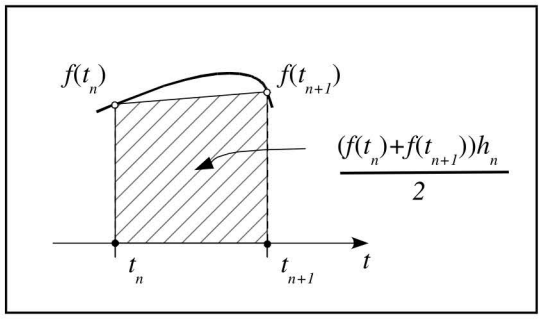
Po obliczeniu całki z powyższego wielomianu po przedziale $[t_n, t_{n+1}]$, otrzymuje się kwadraturę pierwszego rzędu, która umożliwia przybliżone obliczanie wartości całki $I_n(f) = \int_{t_n}^{t_{n+1}} f(t)dt$ z funkcji $f(t)$. Mamy zatem

$$\begin{aligned}\int_{t_n}^{t_{n+1}} L_n(f)dt &= \frac{f(t_n)}{(t_n - t_{n+1})}\int_{t_n}^{t_{n+1}}(t - t_{n+1})dt + \frac{f(t_{n+1})}{(t_{n+1} - t_n)}\int_{t_n}^{t_{n+1}}(t - t_n)dt = \\ &= -\frac{f(t_n)}{2}(t_n - t_{n+1}) + \frac{f(t_{n+1})}{2}(t_{n+1} - t_n) = \left(\frac{f(t_n) + f(t_{n+1})}{2}\right)h_n,\end{aligned}$$

gdzie $h_n = (t_{n+1} - t_n)$ to odległość pomiędzy sąsiednimi punktami węzłowymi. Otrzymana kwadratura $Q_n(f) = h_n(f(t_n) + f(t_{n+1}))/2$ nosi nazwę *wzoru* lub *kwadratury trapezów* przez wzgląd na to, iż pole pod wykresem funkcji $f(t)$, jest przybliżane polem trapezu o wierzchołkach w punktach $f(t_n)$ oraz $f(t_{n+1})$ i podstawie o długości h_n. Sposób numerycznego obliczania wartości całki $I_n(f)$ przy pomocy kwadratury $Q_n(f)$ zobrazowano na rysunku 3.1.

Ze względu na fakt aproksymacji funkcji podcałkowej wielomianem interpolacyjnym pierwszego stopnia, w przypadku ogólnym pomiędzy całką $I_n(f)$ a kwadraturą trapezów skonstruowaną w oparciu o wartości funkcji $f(t)$ w punktach węzłowych t_n i t_{n+1}, zachodzi następujący związek

$$I_n(f) \approx Q_n(f).$$



Rysunek 3.1: Przybliżenie wartości całki $I_n(f)$ poprzez kwadraturę trapezów $Q_n(f)$

Z zależności tej wynika wprost, iż kwadratura $Q_n(f)$ pozwala na obliczenie wartości całki z funkcji $f(t)$, które jest obarczone pewnym błędem, definiowanym jako różnica wartości całki i jej przybliżenia liczonego za pomocą kwadratury, co zapisujemy jako

$$\epsilon_n(f) \triangleq I_n(f) - Q_n(f). \tag{3.2}$$

Dokładną postać wyrażenia na błąd $\epsilon_n(f)$ można wyznaczyć poprzez odpowiednie rozwinięcie funkcji podcałkowej $f(t)$ w *szereg Taylora*‡ (patrz [39]) względem punktu $\hat{t}_n = t_n + (t_{n+1} - t_n)/2$, będącego środkiem przedziału całkowania $[t_n, t_{n+1}]$. Rozpiszmy zatem $I_n(f)$ jako całkę z rozwinięcia funkcji $f(t)$ w szereg Taylora względem tego punktu jako

$$I_n(f) = \int_{t_n}^{t_{n+1}} f(t)dt \overset{(*)}{=} \sum_{l=0}^{K-1} \frac{f^{(l)}(\hat{t}_n)}{l!} \int_{t_n}^{t_{n+1}} (t - \hat{t}_n)^l dt + \frac{f^{(K)}(\xi_n)}{K!} \int_{t_n}^{t_{n+1}} (t - \hat{t}_n)^K dt, \tag{3.3}$$

gdzie ξ_n to pewien punkt leżący w przedziale $[t_n, t_{n+1}]$. W powyższym wzorze pojawia się człon w postaci całki z wyrażenia $(t - \hat{t}_n)^l$, którego postać można wyznaczyć

‡ *Szereg Taylora.*
Jeżeli funkcja $f(t)$, określona na przedziale $[a, b]$, posiada K ciągłych i ograniczonych pochodnych, to funkcję tę można przedstawić względem pewnego punktu $t_0 \in [a, b]$ w postaci następującego szeregu

$$f(t) = \sum_{l=0}^{K-1} \frac{f^{(l)}(t_0)}{l!}(t - t_0)^l + \frac{f^{(K)}(\xi)}{K!}(t - t_0)^K,$$

gdzie ξ leży pomiędzy t_0 i t.

*Należy tutaj zwrócić uwagę na fakt, iż w rozwinięciu funkcji w szereg Taylora ξ jest zależne od zmiennej t. Prawdziwość wyrażenia (3.3) została dowiedziona dla przykładu w pracy [7].

obliczając wartość tej całki w sposób pokazany poniżej

$$\begin{aligned}\int_{t_n}^{t_{n+1}}(t-\hat{t}_n)^l dt &= \frac{1}{l+1}\left[(t_{n+1}-\hat{t}_n)^{l+1}-(t_n-\hat{t}_n)^{l+1}\right]= \\ &= \frac{1}{l+1}\left[\left(\frac{t_{n+1}-t_n}{2}\right)^{l+1}-(-1)^{l+1}\left(\frac{t_{n+1}-t_n}{2}\right)^{l+1}\right]= \\ &= \begin{cases}\dfrac{(t_{n+1}-t_n)^{l+1}}{2^l(l+1)} & \text{dla } l \text{ parzystych,}\\ \\ 0 & \text{dla } l \text{ nieparzystych.}\end{cases}\end{aligned} \tag{3.4}$$

Uwzględniając uzyskany w ten sposób wynik (3.4) we wzorze (3.3), otrzymujemy szukaną postać całki z rozwinięcia funkcji $f(t)$ w szereg Taylora względem punktu $\hat{t}_n$

$$I_n(f)=\int_{t_n}^{t_{n+1}} f(t)dt=\sum_{l=0}^{K/2-1}\frac{f^{(2l)}(\hat{t}_n)}{(2l+1)!2^{2l}}h_n^{2l+1}+\frac{f^{(K)}(\xi_n)}{(K+1)!2^K}h_n^{K+1}. \tag{3.5}$$

W podobny sposób rozpiszmy wzór trapezów $Q_n(f)=h_n(f(t_n)+f(t_{n+1}))/2$, korzystając z rozwinięcia funkcji $f(t)$ w szereg Taylora względem punktu środkowego dla przedziału $[t_n,t_{n+1}]$. Otrzymujemy wówczas następujące wyrażenie

$$\begin{aligned}Q_n(f) &= h_n\left(\frac{f(t_n)+f(t_{n+1})}{2}\right)=\frac{h_n}{2}\left[\sum_{l=0}^{K-1}\frac{f^{(l)}(\hat{t}_n)}{l!}\left((t_n-\hat{t}_n)^l+(t_{n+1}-\hat{t}_n)^l\right)\right]+ \\ &+ \frac{h_n}{2}\left[\frac{f^{(K)}(\xi_n)}{K!}\left((t_n-\hat{t}_n)^K+(t_{n+1}-\hat{t}_n)^K\right)\right]= \\ &= \frac{h_n}{2}\left[\sum_{l=0}^{K-1}\frac{f^{(l)}(\hat{t}_n)}{l!}\left(\left(\frac{t_{n+1}-t_n}{2}\right)^l+(-1)^l\left(\frac{t_{n+1}-t_n}{2}\right)^l\right)\right]+ \\ &+ \frac{h_n}{2}\left[\frac{f^{(K)}(\xi_n)}{K!}\left(\left(\frac{t_{n+1}-t_n}{2}\right)^K+(-1)^K\left(\frac{t_{n+1}-t_n}{2}\right)^K\right)\right]= \\ &= \sum_{l=0}^{K/2-1}\frac{f^{(2l)}(\hat{t}_n)}{(2l)!2^{2l}}h_n^{2l+1}+\frac{f^{(K)}(\xi_n)}{K!2^K}h_n^{K+1}.\end{aligned} \tag{3.6}$$

Po podstawieniu wyrażeń, które uzyskano w wyniku przekształceń opisanych zależnościami (3.5) oraz (3.6) do wzoru na postać błędu numerycznego obliczania całki $I_n(f)$ za pomocą kwadratury $Q_n(f)$ (patrz wzór (3.2)), otrzymujemy ostateczną postać wyrażenia na błąd $\epsilon_n(f)$. Wielkość błędu zależy od sumy ważonej parzystych pochodnych funkcji $f(t)$, liczonych w punktach $\hat{t}_n$ oraz ξ_n

$$\epsilon_n(f)=-\left(\sum_{l=1}^{K/2-1}\frac{f^{(2l)}(\hat{t}_n)2l}{2^{2l}(2l+1)!}h_n^{2l+1}+\frac{f^{(K)}(\xi_n)K}{2^K(K+1)!}h_n^{K+1}\right). \tag{3.7}$$

W szczególności, po podstawieniu za $K=2$, powyższy wzór przyjmie dobrze znaną z literatury (patrz [7, 32, 39, 40]) postać wyrażenia

$$\epsilon_n(f)=I_n(f)-Q_n(f)=-\frac{f^{(2)}(\xi_n)}{12}h_n^3,$$

gdzie $\xi_n\in[t_n,t_{n+1}]$.

W dotychczasowych rozważaniach wyznaczyliśmy postać wzoru na błąd numerycznego obliczania całki z funkcji $f(t)$ na przedziale $[t_n, t_{n+1}]$, który jest jedynie częścią całego przedziału $[a, b]$ określoności funkcji podcałkowej $f(t)$. Stosując analogiczny tok rozumowania w odniesieniu do wszystkich $N-1$ podprzedziałów, które są jednoznacznie wyznaczone dla poszczególnych kwadratur $Q_n(f)$ poprzez zbiór punktów węzłowych $\{t_n : n = 0, 1, ..., N\}$, a także korzystając z elementarnych reguł całkowania, można zapisać

$$I(f) = \int_a^b f(t)dt = \sum_{n=0}^{N-1} \int_{t_n}^{t_{n+1}} f(t)dt \approx \sum_{n=0}^{N-1} Q_n(f),$$

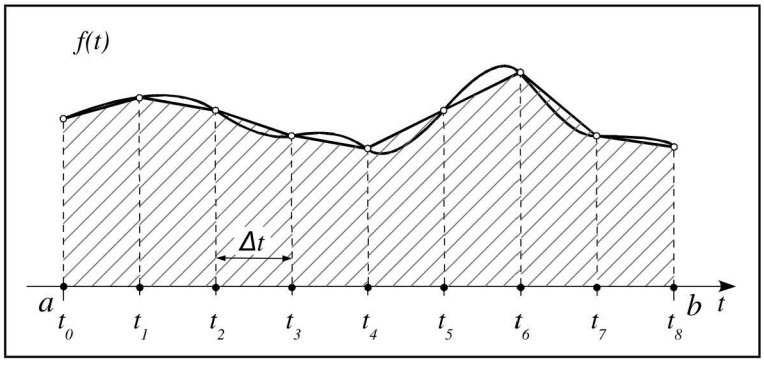
gdzie prawa strona powyższej zależności jest złożoną kwadraturą trapezów. Przyjmijmy dodatkowo, że punkty węzłowe t_n rozłożone są w sposób równomierny, tzn. $h_n = \Delta t$ dla $n = 0, 1, ..., N-1$ i oznaczmy uzyskaną w ten sposób kwadraturę jako

$$Q_N(f) = \sum_{n=0}^{N-1} Q_n(f) = \Delta t \sum_{n=0}^{N-1} \left(\frac{f(t_n) + f(t_{n+1})}{2} \right). \tag{3.8}$$

Wówczas zgodnie ze wzorem (3.7), błąd obliczania całki $I(f)$ przy pomocy złożonej kwadratury numerycznej $Q_N(f)$ o równomiernie rozłożonych węzłach, określa zależność postaci

$$\begin{aligned} \epsilon_N(f) &= I(f) - Q_N(f) = \\ &= -\sum_{n=0}^{N-1} \left(\sum_{l=1}^{K/2-1} \frac{f^{(2l)}(\hat{t}_n) 2l}{2^{2l}(2l+1)!} \Delta t^{2l+1} + \frac{f^{(K)}(\xi_n) K}{2^K (K+1)!} \Delta t^{K+1} \right), \end{aligned} \tag{3.9}$$

gdzie punkty $\hat{t}_n$ wyznaczamy według reguły $\hat{t}_n = t_n + \Delta t/2$. Wzór ten stanowi sumę błędów przybliżonego obliczania wartości poszczególnych całek $I_n(f)$ poprzez kwadratury $Q_n(f)$. Na rysunku 3.2 zobrazowano sposób numerycznego obliczania wartości całki z funkcji $f(t)$ przy użyciu złożonej kwadratury trapezów $Q_N(f)$ o równomiernie rozłożonych węzłach.

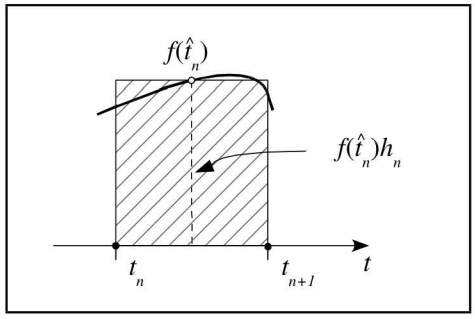


Rysunek 3.2: Numeryczne obliczanie całki za pomocą złożonej kwadratury trapezów o równomiernie rozłożonych węzłach dla przypadku $N = 8$ punktów węzłowych

Z pracy [7] wiadomo, iż kwadratury parzystych rzędów r mogą zachowywać się jak kwadratury wyższego rzędu, tzn. dają dokładne rezultaty dla wielomianów co najwyżej stopnia $(r+1)$-szego. Taka sytuacja ma miejsce w przypadku kwadratury prostokątów (będącej kwadraturą rzędu $r = 0$, tzn. kwadraturą dokładną dla wielomianów stopnia $r = 0$) o punktach węzłowych wyznaczonych w środkach przedziałów $[t_n, t_{n+1}]$, tj. w punktach definiowanych jako $\hat{t}_n = t_n + (t_{n+1} - t_n)/2$. Aby tego dowieść, wystarczy rozpisać wyrażenie (3.5) w następujący sposób

$$I_n(f) = \int_{t_n}^{t_{n+1}} f(t)dt = f(\hat{t}_n)h_n + \sum_{l=1}^{K/2-1} \frac{f^{(2l)}(\hat{t}_n)}{(2l+1)!2^{2l}} h_n^{2l+1} + \frac{f^{(K)}(\xi_n)}{(K+1)!2^K} h_n^{K+1},$$

gdzie człon $f(\hat{t}_n)h_n$ jest kwadraturą prostokątów przybliżającą wartość całki $I_n(f)$ poprzez pole prostokąta o podstawie h_n i wysokości $f(\hat{t}_n)$ (patrz rysunek 3.3). Oznaczając



Rysunek 3.3: Przybliżenie wartości całki $I_n(f)$ przez kwadraturę prostokątów o węźle umiejscowionym w środku przedziału całkowania

przez $\hat{Q}_n(f) = f(\hat{t}_n)h_n$, otrzymujemy natychmiast na podstawie powyższej zależności postać błędu numerycznego obliczania całki $I_n(f)$ przy pomocy tej kwadratury

$$\hat{\epsilon}_n(f) = I_n(f) - \hat{Q}_n(f) = \sum_{l=1}^{K/2-1} \frac{f^{(2l)}(\hat{t}_n)}{(2l+1)!2^{2l}} h_n^{2l+1} + \frac{f^{(K)}(\xi_n)}{(K+1)!2^K} h_n^{K+1}.$$

Złożoną kwadraturę prostokątów, zbudowaną zgodnie z regułą zademonstrowaną na rysunku 3.3, można wówczas zapisać jako sumę kwadratur $\hat{Q}_n(f)$, liczoną po wszystkich podprzedziałach $[t_n, t_{n+1}]$ przedziału całkowania. W przypadku szczególnym, gdy wszystkie podprzedziały $[t_n, t_{n+1}]$ mają taką samą długość, tzn. $h_n = (t_{n+1} - t_n) = \Delta t$ dla $n = 0, 1, ..., N-1$, wyrażenie to przyjmie postać wzoru

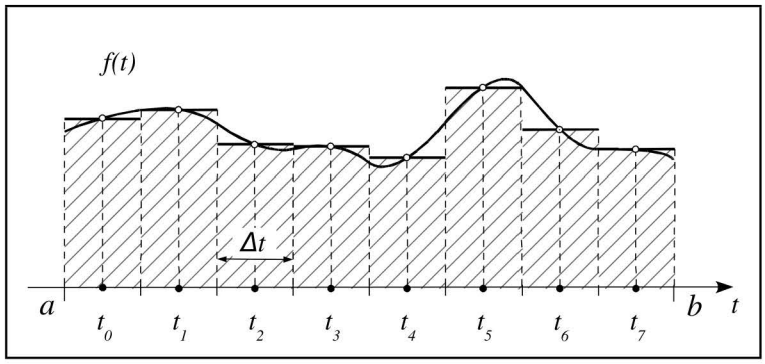
$$\hat{Q}_N(f) = \sum_{n=0}^{N-1} \hat{Q}_n(f) = \Delta t \sum_{n=0}^{N-1} f(\hat{t}_n). \tag{3.10}$$

Dla dowolnie wybranej funkcji podcałkowej prawdziwa jest jedynie zależność postaci $I(f) \approx \hat{Q}_N(f)$. Dokładną wartość błędu numerycznego obliczania całki $I(f)$ przy użyciu złożonej kwadratury prostokątów o równomiernie rozłożonych węzłach, które przypadają w środkach podprzedziałów postaci $[t_n, t_{n+1}]$, można wyznaczyć za pomocą

następującej zależności

$$\begin{aligned} \hat{\epsilon}_N(f) &= I(f) - \hat{Q}_N(f) = \\ &= \sum_{n=0}^{N-1} \left(\sum_{l=1}^{K/2-2} \frac{f^{(2l)}(\hat{t}_n)}{(2l+1)!2^{2l}} \Delta t^{2l+1} + \frac{f^{(K)}(\xi_i)}{(K+1)!2^K} \Delta t^{K+1} \right). \end{aligned} \tag{3.11}$$

Sposób numerycznego obliczania całki z funkcji $f(t)$ przy pomocy tak skonstruowanej kwadratury, zobrazowano na rysunku 3.4. Z analizy wyrażenia (3.11) wynika, iż błąd kwadratury $\hat{Q}_N(f)$ zależy od wartości parzystych pochodnych funkcji $f(t)$, a zatem, podobnie jak kwadratura $Q_N(f)$, jest to kwadratura rzędu $r = 1$.



Rysunek 3.4: Numeryczne obliczanie całki za pomocą złożonej kwadratury prostokątów o równomiernie rozłożonych węzłach i punktach węzłowych przypadających w środkach przedziałów o długościach Δt, dla przypadku $N = 8$ punktów węzłowych

Dotychczas wykazano, iż złożone kwadratury trapezów o równomiernie rozłożonych węzłach (3.8), oraz złożone kwadratury prostokątów (3.10) o węzłach umiejscowionych w środkach podprzedziałów $[t_n, t_{n+1}]$, cechuje błąd zależny od pochodnych parzystych rzędów. Zatem kwadratury te są kwadraturami rzędu pierwszego, tzn. dają dokładne wyniki dla funkcji podcałkowych będących wielomianami co najwyżej pierwszego stopnia.

W rozdziale 2, w którym zamieszczone zostały definicje oraz podstawowe własności całkowych i dyskretnych przekształceń trygonometrycznych, wykazano również, że przekształcenia dyskretne w przypadku ogólnym pozwalają na przybliżone obliczanie wartości przekształceń w postaci całkowej. W dalszej części bieżącej sekcji dyskretne jednowymiarowe przekształcenia trygonometryczne przeanalizowane zostaną pod kątem teorii całkowania numerycznego. Jako pierwszy rozważony zostanie przypadek dyskretnego przekształcenia Fouriera, a następnie przypadki dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju.

Dyskretne przekształcenie Fouriera

Zgodnie z rozdziałem 2 dyskretne jednowymiarowe N-punktowe przekształcenie Fouriera $X_N(k)$, definiowane jako wyrażenie (2.8a), pozwala na przybliżone obliczanie całkowego przekształcenia Fouriera $X(\omega_k)$ postaci (2.1), dla dyskretnych wartości pulsacji $\omega_k = (2\pi/T)k$, gdzie parametr $k = 0, \pm 1, ..., \pm\frac{N}{2}$. Wówczas w przypadku ogólnym

zachodzi jedynie następująca zależność

$$X(\omega_k) \approx TX_N(k)$$

dla $k = 0, \pm 1, ..., \pm\frac{N}{2}$. Korzystając ze wzoru (2.8a), a także mając na uwadze zastosowaną regułę doboru próbek sygnału w czasie postaci $t_n = n\Delta t$, z przyjętym krokiem dyskretyzacji $\Delta t = T/N$ dla $n = 0, 1, ..., N-1$, możemy rozpisać prawą stronę powyższej zależności jako

$$TX_N(k) = \Delta t \sum_{n=0}^{N-1} x(t_n) e^{-i\omega_k t_n} = \Delta t \sum_{n=0}^{N-1} f(t_n, k). \tag{3.12}$$

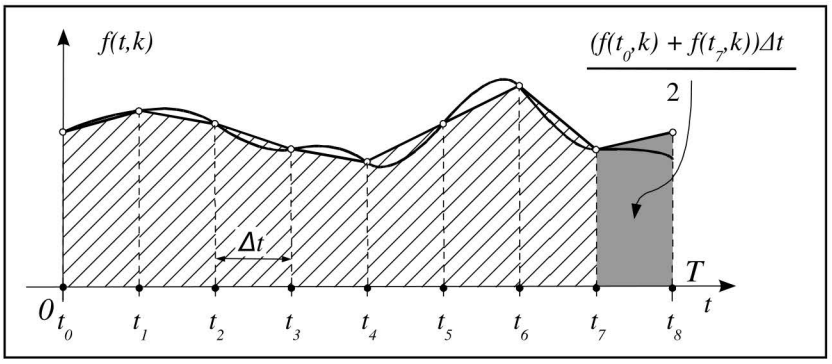
Jeżeli rozpatrywać tę zależność w kategoriach kwadratur całkowania numerycznego to zauważymy, iż wzór ten jest wzorem prostokątów o równomiernie rozłożonych węzłach, umiejscowionych na początkach przedziałów $[t_n, t_{n+1}]$, o stałej długości Δt. Zatem wzór ten umożliwia przybliżone obliczanie całki z funkcji postaci $f(t,k) = x(t)e^{-i\omega_k t}$ na przedziale $[0, T]$, gdzie funkcja podcałkowa $f(t,k)$ jest dodatkowo sparametryzowana poprzez czynnik k, który związany jest z pulsacją ω_k. W rezultacie, mamy tutaj do czynienia z przypadkiem N całek (innej dla każdej wartości parametru k), których wartości obliczane są w sposób numeryczny przy pomocy kwadratur prostokątów. Należy pamiętać, że funkcja podcałkowa $f(t,k)$ przyjmuje wartości zespolone. W takiej sytuacji jej części rzeczywistą $Re\{f(t,k)\} = x(t)cos(\omega_k t)$ i urojoną $Im\{f(t,k)\} = -ix(t)sin(\omega_k t)$ należy traktować jako przypadki dwóch osobnych funkcji.

Ponieważ przyjęta reguła dyskretyzacji w czasie zakłada umiejscowienie punktów węzłowych kwadratury w początkach przedziałów $[t_n, t_{n+1}]$, to można wykazać, że kwadratura (3.12) jest kwadraturą stopnia zero, tzn. daje dokładne rezultaty dla wielomianów stopnia co najwyżej równego zeru. Wyrażenie (3.12) można jednak rozpisać jako

$$\begin{aligned} X_N(k) \;=\; & \Delta t \left(\frac{f(t_0,k) + f(t_1,k)}{2} + \frac{f(t_1,k) + f(t_2,k)}{2} + ... \right. \\ & \left. ... + \frac{f(t_{N-2},k) + f(t_{N-1},k)}{2} + \frac{f(t_0,k) + f(t_{N-1},k)}{2} \right). \end{aligned} \tag{3.13}$$

Porównując tak rozpisany wzór (3.13) z kwadraturą (3.8) zauważymy, iż różnią się one jedynie ostatnim czynnikiem, którego wpływ oznaczono na rysunku 3.5 szarym i jednolitym obszarem. Zatem wprowadzając do wyrażenia (3.13) niewielką modyfikację postaci $f(t_0,k) = (f(t_0,k) + f(t_N,k))/2$, jesteśmy w stanie wyrażenie to przekształcić do postaci złożonej kwadratury trapezów określonej wzorem (3.8), otrzymując w ten sposób kwadraturę rzędu pierwszego.

Jak już wcześniej wspomniano funkcja podcałkowa $f(t,k)$ jest funkcją zespoloną postaci $f(t,k) = x(t)cos(\omega_k t) - ix(t)sin(\omega_k t)$, którą rozpatrujemy jako przypadek dwóch osobnych funkcji $x(t)cos(\omega_k t)$ oraz $-ix(t)sin(\omega_k t)$. Wówczas dyskretne przekształcenie Fouriera należy traktować jako zespół kwadratur pozwalających na przybliżone obliczanie całek postaci $\int_0^T x(t)cos(\omega_k t)dt$ oraz $-i\int_0^T x(t)sin(\omega_k t)dt$. Ponieważ dla przyjętych dyskretnych wartości pulsacji $\omega_k = k\Delta\omega$, gdzie $\Delta\omega = 2\pi/T$ funkcja $-ix(t)sin(\omega_k t)$ na obu krańcach przedziału całkowania $[0, T]$ przyjmuje identyczne wartości, równe zeru, tzn. $-ix(t_0)sin(\omega_k t_0) = -ix(t_N)sin(\omega_k t_N) = 0$, to kwadratura numeryczna o postaci $-i\Delta t \sum_{n=0}^{N-1} x(t_n)sin(\omega_k t_n)$ jest kwadraturą trapezów. Zatem wskazana modyfikacja



Rysunek 3.5: Sposób przybliżania wartości całki według kwadratury opisanej wzorem (3.13) dla przypadku $N = 8$ punktów węzłowych

w zbiorze próbek transformowanego sygnału jest wymagana wyłącznie dla następującej kwadratury $\Delta t \sum_{n=0}^{N-1} x(t_n) cos(\omega_k t_n)$.

Z przeprowadzonego toku rozumowania wynika następujący wniosek. Otóż dyskretne przekształcenie Fouriera (z dokładnością do stałej T) stanowi zespół (określonych względem parametru $k = 0, 1, ..., N-1$) kwadratur prostokątów o równomiernie rozłożonych węzłach, umożliwiających numeryczne obliczanie całek z zespolonych funkcji $f(t,k) = x(t)e^{-i\omega_k t}$ na przedziale $[0,T]$, które odpowiadają wartościom $X(\omega_k)$ całkowego przekształcenia Fouriera dla $k = 0, \pm 1, ..., \pm\frac{N}{2}$. Wprowadzając następującą modyfikację

$$f(t_0,k) = \frac{f(t_0,k) + f(t_N,k)}{2} = \frac{x(t_0)e^{-i\frac{2\pi}{N}k0} + x(t_N)e^{-i\frac{2\pi}{N}kN}}{2} = \frac{x(t_0) + x(t_N)}{2},$$

równoważną wyrażeniu

$$x(t_0) = \frac{x(t_0) + x(t_N)}{2}, \tag{3.14}$$

otrzymujemy w postaci przekształcenia DFT, zespół kwadratur pierwszego rzędu, tj. kwadratur trapezów definiowanych zgodnie z zależnością (3.8).

Dyskretne przekształcenia kosinusowe i sinusowe drugiego oraz czwartego rodzaju

Drugi przypadek stanowią kosinusowe i sinusowe przekształcenie Fouriera. Tutaj funkcje podcałkowe przyjmują wyłącznie wartości rzeczywiste. Zatem mając na względzie przypadek dyskretnego przekształcenia Fouriera, oraz regułę doboru próbek w dziedzinie czasu charakterystyczną dla dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju, która zakłada dobór próbek w punktach $\hat{t}_n = (n + 1/2)\Delta t$ będących środkami przedziałów $[t_n, t_{n+1}]$ o długościach Δt, można jednoznacznie stwierdzić, iż przekształcenia te (z dokładnością do stałej Tp_k lub T) stanowią zespoły kwadratur prostokątów (3.10) pierwszego rzędu, dla numerycznego obliczania całkowych przekształceń kosinusowych i sinusowych (patrz rozdział 2).

3.2. Dyskretne przekształcenia dwuwymiarowe jako kubatury numeryczne

W tej części rozdziału rozpatrzony zostanie przypadek dwuwymiarowych przekształceń trygonometrycznych. W tym celu zakładamy, że dana jest funkcja $f(t,s)$ dwóch zmiennych $t,s \in \mathcal{R}$, określona na obszarze prostokątnym $[a_1,b_1]\times[a_2,b_2]$, która posiada ograniczone pochodne cząstkowe i mieszane aż do K-tego rzędu (K jest parzyste) i może przyjmować wyłącznie wartości rzeczywiste. Zakładamy dodatkowo następujący podział obszaru całkowania na N na M rozłącznych obszarów prostokątnych postaci $[t_n,t_{n+1}]\times[s_m,s_{m+1}]$ dla $n=0,1,...,N-1$ i $m=0,1,...,M-1$, gdzie $a_1=t_0<t_1<...<t_N=b_1$ oraz $a_2=s_0<s_1<...<s_M=b_2$. Wówczas zgodnie z pracą [16] funkcję tę można przedstawić względem punktu $(\hat{t}_n,\hat{s}_m)$ środkowego podprzedziału $[t_n,t_{n+1}]\times[s_m,s_{m+1}]$ w postaci szeregu Taylora, jako następującą sumę

$$\begin{aligned} f(t,s) &= \sum_{l=0}^{K-1}\left(\sum_{i=0}^{l}\frac{(t-\hat{t}_n)^{l-i}(s-\hat{s}_m)^i}{i!(l-i)!}\frac{\partial^l f(\hat{t}_n,\hat{s}_m)}{\partial t^{l-i}\partial s^i}\right)+ \\ &+ \sum_{i=0}^{K}\frac{(t-\hat{t}_n)^{K-i}(s-\hat{s}_m)^i}{i!(K-i)!}\frac{\partial^K f(\epsilon_n,\eta_m)}{\partial t^{K-i}\partial s^i}, \end{aligned}$$

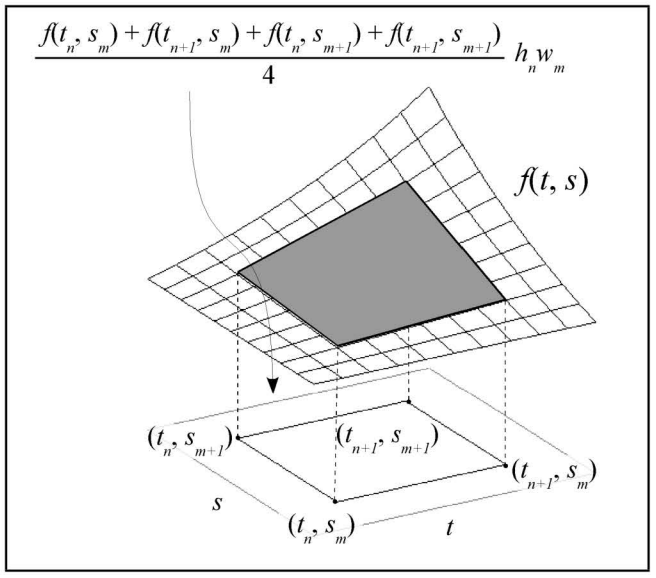
gdzie punkt $(\epsilon_n,\eta_m)\in[t_n,t_{n+1}]\times[s_m,s_{m+1}]$. Następnie obliczmy całkę z powyższego wyrażenia po przedziale $[t_n,t_{n+1}]\times[s_m,s_{m+1}]$ i oznaczamy ją symbolicznie jako $I^{2D}_{n,m}(f)$. Korzystając z analogicznych jak w przypadku jednowymiarowym przekształceń można wykazać, że całka ta przyjmie postać wyrażenia

$$\begin{aligned} I^{2D}_{n,m}(f) &= \int_{t_n}^{t_{n+1}}\int_{s_m}^{s_{m+1}} f(t,s)dtds = \\ &= \sum_{l=0}^{K/2-1}\left(\sum_{i=0}^{l}\frac{h_n^{2(l-i)+1}w_m^{2i+1}}{(2i+1)!(2(l-i)+1)!2^{2l}}\frac{\partial^{2l} f(\hat{t}_n,\hat{s}_m)}{\partial t^{2(l-i)}\partial s^{2i}}\right)+ \\ &+ \sum_{i=0}^{K/2}\frac{h_n^{K-2i+1}w_m^{2i+1}}{(2i+1)!(K-2i+1)!2^K}\frac{\partial^K f(\epsilon_n,\eta_m)}{\partial t^{K-2i}\partial s^{2i}}, \end{aligned} \tag{3.15}$$

gdzie $h_n=t_{n+1}-t_n$ oraz $w_m=s_{m+1}-s_m$. Wprowadźmy następujące wyrażenie umożliwiające przybliżone obliczenie wartości całki $I^{2D}_{n,m}(f)$ na bazie wartości funkcji $f(t,s)$ pobranych w czterech punktach węzłowych, które są wierzchołkami prostokątnego obszaru $[t_n,t_{n+1}]\times[s_m,s_{m+1}]$

$$Q^{2D}_{n,m}(f)=\left(\frac{f(t_n,s_n)+f(t_{n+1},s_m)+f(t_n,s_{m+1})+f(t_{n+1},s_{m+1})}{4}\right)h_n w_m. \tag{3.16}$$

Wyrażenie to zgodnie z pracą [107] nosi nazwę *kubatury całkowania numerycznego*. Sposób przybliżenia wartości całki poprzez kubaturę $Q^{2D}_{n,m}(f)$ zademonstrowany został na rysunku 3.6. Wartość całki z funkcji $f(t,s)$ jest tutaj przybliżana objętością bryły



Rysunek 3.6: Przybliżone obliczanie całki $I^{2D}_{n,m}(f)$ za pomocą kubatury $Q^{2D}_{n,m}(f)$

powstałej pod wykresem funkcji aproksymującej postaci

$$\begin{aligned} L_{n,m}(f) &= f(t_n, s_m) + \left(\frac{f(t_{n+1}, s_m) - f(t_n, s_m)}{h_n}\right)(t - t_n) + \\ &+ \left(\frac{f(t_n, s_{m+1}) - f(t_n, s_m)}{w_m}\right)(s - s_m) + \left(\frac{f(t_{n+1}, s_{m+1}) + f(t_n, s_m)}{h_n w_m} - \right. \\ &- \left. \frac{f(t_{n+1}, s_m) + f(t_n, s_{m+1})}{h_n w_m}\right)(t - t_n)(s - s_n), \end{aligned}$$

którą oznaczono na rysunku 3.6 szarym obszarem. W przypadku ogólnym zachodzi jedynie następująca zależność

$$I^{2D}_{n,m}(f) \approx Q^{2D}_{n,m}(f),$$

co oznacza, że wartość całki z funkcji $f(t, s)$ na przedziale $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$ jest obliczana za pomocą $Q^{2D}_{n,m}(f)$ z pewnym błędem, definiowanym jako

$$\epsilon^{2D}_{n,m}(f) \triangleq I^{2D}_{n,m}(f) - Q^{2D}_{n,m}(f). \tag{3.17}$$

Stosując rozwinięcie funkcji $f(t, s)$ w szereg Taylora względem punktu leżącego w środku obszaru $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$, można przedstawić kubaturę $Q^{2D}_{n,m}(f)$ w postaci sumy ważonej jej pochodnych cząstkowych jako

$$\begin{aligned} Q^{2D}_{n,m}(f) &= \sum_{l=0}^{K/2-1} \left(\sum_{i=0}^{l} \frac{h_n^{2(l-i)+1} w_m^{2i+1}}{(2i)!(2(l-i))! 2^{2l}} \frac{\partial^{2l} f(\hat{t}_n, \hat{s}_m)}{\partial t^{2(l-i)} \partial s^{2i}} \right) + \\ &+ \sum_{i=0}^{K/2} \frac{h_n^{K-2i+1} w_m^{2i+1}}{(2i)!(K-2i)! 2^K} \frac{\partial^K f(\epsilon_n, \eta_m)}{\partial t^{K-2i} \partial s^{2i}} \end{aligned} \tag{3.18}$$

dla $(\epsilon_n, \eta_m) \in [t_n, t_{n+1}] \times [s_m, s_{m+1}]$. Wówczas po podstawieniu wyrażenia (3.15) oraz (3.18) do wzoru definiującego błąd (3.17) kubatury $Q^{2D}_{n,m}(f)$, otrzymamy następującą postać tego błędu

$$\begin{aligned} \epsilon^{2D}_{n,m}(f) &= -\sum_{l=1}^{K/2-1}\left(\sum_{i=0}^{l}\frac{(2l(2i+1)-4i^2)h_n^{2(l-i)+1}w_m^{2i+1}}{(2i+1)!(2(l-i)+1)!2^{2l}}\frac{\partial^{2l}f(\hat{t}_n,\hat{s}_m)}{\partial t^{2(l-i)}\partial s^{2i}}\right)- \\ &- \sum_{i=0}^{K/2}\frac{(K(2i+1)-4i^2)h_n^{K-2i+1}w_m^{2i+1}}{(2i+1)!(K-2i+1)!2^K}\frac{\partial^K f(\epsilon_n,\eta_m)}{\partial t^{K-2i}\partial s^{2i}}. \end{aligned} \tag{3.19}$$

Z analizy powyższej zależności wynika, iż błąd numerycznego obliczania całki $I^{2D}_{n,m}(f)$ przy pomocy kubatury $Q^{2D}_{n,m}(f)$, zależy wprost proporcjonalnie od powierzchni obszaru całkowania oraz pochodnych cząstkowych funkcji $f(t,s)$. Zatem, o ile pochodne cząstkowe są ograniczone, to błąd ten można dowolnie redukować poprzez zmniejszanie powierzchni $h_n w_n$ obszaru całkowania $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$.

Kubatura $Q^{2D}_{n,m}(f)$ jest określona na pewnym podprzedziale $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$ całego przedziału $[a_1, b_1] \times [a_2, b_2]$. Zatem, podobnie jak w przypadku jednowymiarowym, można na jej podstawie zbudować *kubaturę złożoną*, która umożliwi numeryczne obliczenie wartości całki postaci

$$I^{2D}(f) = \int_{a_1}^{b_1}\int_{a_2}^{b_2} f(t,s)dtds.$$

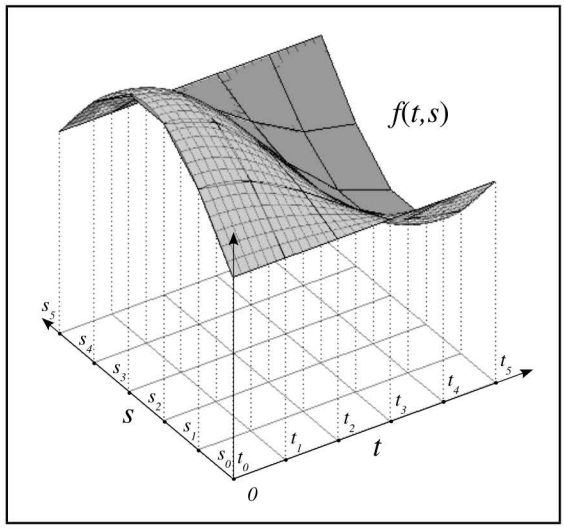
Kubatura złożona będzie wówczas sumą wartości kubatur prostych $Q^{2D}_{n,m}(f)$, liczoną po wszystkich podprzedziałach $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$ i przyjmie postać wyrażenia

$$\begin{aligned} Q^{2D}_{N\cdot M}(f) &= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} Q^{2D}_{n,m}(f) = \\ &= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(\frac{f(t_n,s_n)+f(t_{n+1},s_m)+f(t_n,s_{m+1})+f(t_{n+1},s_{m+1})}{4}\right)h_n w_m. \end{aligned}$$

Zakładając dodatkowo, że dla każdego $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$ parametry $h_n = \Delta t$ oraz $w_m = \Delta s$ są stałe i niezależne od wartości n i m, tzn. wszystkie obszary $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$ mają taką samą powierzchnię równą $\Delta t \Delta s$, to powyższa kubatura przyjmie postać *równomiernej kubatury złożonej*, określonej zależnością

$$Q^{2D}_{N\cdot M}(f) = \Delta t \Delta s \sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(\frac{f(t_n,s_n)+f(t_{n+1},s_m)+f(t_n,s_{m+1})+f(t_{n+1},s_{m+1})}{4}\right). \tag{3.20}$$

Sposób przybliżonego obliczania wartości całki $I^{2D}(f)$ przy pomocy równomiernej kubatury złożonej $Q^{2D}_{N\cdot M}(f)$ zaprezentowano na rysunku 3.7. Błąd numerycznego obliczania całki przy pomocy równomiernej kubatury złożonej będzie sumą błędów (3.19) kubatur prostych $Q^{2D}_{n,m}(f)$, liczoną po wszystkich podprzedziałach $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$,



Rysunek 3.7: Przybliżone obliczanie całki $I^{2D}(f)$ za pomocą złożonej kubatury równomiernej $Q^{2D}_{N\cdot M}(f)$ dla przypadku $N = M = 5$

tzn.

$$\epsilon^{2D}_{N\cdot M}(f) = I^{2D}(f) - Q^{2D}_{N\cdot M} = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} \epsilon^{2D}_{n,m}(f) = \tag{3.21}$$

$$= -\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left[\sum_{l=1}^{K/2-1}\left(\sum_{i=0}^{l}\frac{(2l(2i+1)-4i^2)\Delta t^{2(l-i)+1}\Delta s^{2i+1}}{(2i+1)!(2(l-i)+1)!2^{2l}}\frac{\partial^{2l} f(\hat{t}_n,\hat{s}_m)}{\partial t^{2(l-i)}\partial s^{2i}}\right) + \right.$$

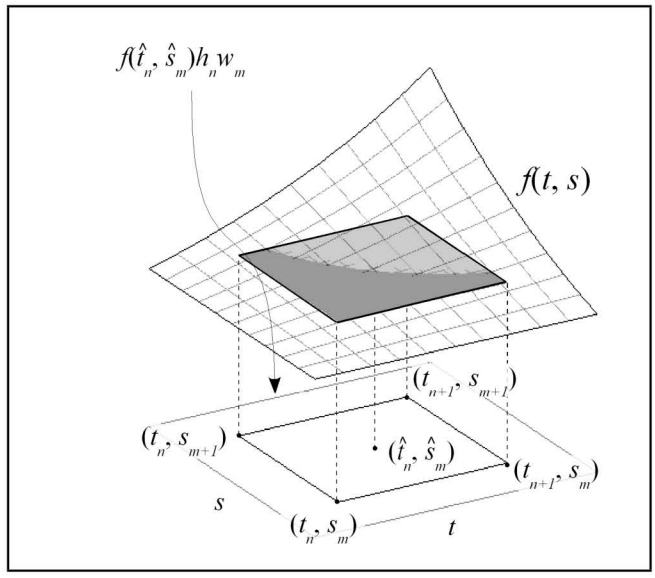
$$\left. + \sum_{i=0}^{K/2}\frac{(K(2i+1)-4i^2)\Delta t^{K-2i+1}\Delta s^{2i+1}}{(2i+1)!(K-2i+1)!2^{K}}\frac{\partial^{K} f(\epsilon_n,\eta_m)}{\partial t^{K-2i}\partial s^{2i}}\right].$$

Wartość błędu $\epsilon^{2D}_{N\cdot M}(f)$ dla kubatury równomiernej zależy zatem wprost proporcjonalnie od wartości parzystych pochodnych cząstkowych funkcji $f(t,s)$ oraz od wielkości parametrów determinujących powierzchnie obszarów, tzn. Δt i Δs, na których oparte zostały kubatury składowe $Q^{2D}_{n,m}(f)$.

W dalszej kolejności rozważony zostanie przykład jeszcze jednej kubatury, która przybliża wartość całki $I^{2D}_{n,m}(f)$ objętością prostopadłościanu o podstawach długości h_n oraz w_m i wysokości będącej wartością funkcji podcałkowej w punkcie środkowym przedziału $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$. Kubaturę taką można wówczas zapisać w postaci następującej zależności

$$\hat{Q}^{2D}_{n,m} = f(\hat{t}_n, \hat{s}_m) h_n w_n,$$

a proces przybliżenia wartości całki $I^{2D}_{n,m}(f)$ za jej pomocą został zobrazowany na rysunku 3.8. Błąd numerycznego obliczania całki $I^{2D}_{n,m}(f)$ przy użyciu powyższej kubatury



Rysunek 3.8: Przybliżone obliczanie całki $I^{2D}_{n,m}(f)$ za pomocą kubatury $\hat{Q}^{2D}_{n,m}(f)$

można łatwo wyznaczyć poprzez odpowiednie rozpisanie zależności (3.15)

$$\begin{aligned} I^{2D}_{n,m}(f) &= \int_{t_n}^{t_{n+1}} \int_{s_m}^{s_{m+1}} f(t,s)dtds = f(\hat{t}_n, \hat{s}_m)h_n w_m + \\ &+ \sum_{l=1}^{K/2-1} \left(\sum_{i=0}^{l} \frac{h_n^{2(l-i)+1} w_m^{2i+1}}{(2i+1)!(2(l-i)+1)!2^{2l}} \frac{\partial^{2l} f(\hat{t}_n, \hat{s}_m)}{\partial t^{2(l-i)} \partial s^{2i}} \right) + \\ &+ \sum_{i=0}^{K/2} \frac{h_n^{K-2i+1} w_m^{2i+1}}{(2i+1)!(K-2i+1)!2^K} \frac{\partial^K f(\epsilon_n, \eta_m)}{\partial t^{K-2i} \partial s^{2i}}. \end{aligned} \tag{3.22}$$

Z powyższej zależności otrzymujemy natychmiast

$$I^{2D}_{n,m}(f) = \hat{Q}^{2D}_{n,m}(f) + \hat{\epsilon}^{2D}_{n,m}(f),$$

gdzie $\hat{\epsilon}^{2D}_{n,m}(f)$ to wyrażenie na błąd kubatury $\hat{Q}^{2D}_{n,m}(f)$, definiowane według (3.22) jako

$$\begin{aligned} \hat{\epsilon}^{2D}_{n,m}(f) &\triangleq I^{2D}_{n,m}(f) - \hat{Q}^{2D}_{n,m}(f) = \\ &= \sum_{l=1}^{K/2-1} \left(\sum_{i=0}^{l} \frac{h_n^{2(l-i)+1} w_m^{2i+1}}{(2i+1)!(2(l-i)+1)!2^{2l}} \frac{\partial^{2l} f(\hat{t}_n, \hat{s}_m)}{\partial t^{2(l-i)} \partial s^{2i}} \right) + \\ &+ \sum_{i=0}^{K/2} \frac{h_n^{K-2i+1} w_m^{2i+1}}{(2i+1)!(K-2i+1)!2^K} \frac{\partial^K f(\epsilon_n, \eta_m)}{\partial t^{K-2i} \partial s^{2i}}. \end{aligned}$$

Złożona kubatura oparta o dobór współczynników węzłowych w środkach przedziałów $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$, przyjmie postać sumy kubatur prostych $\hat{Q}^{2D}_{n,m}(f)$, liczonej po

parametrach $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$, tzn.

$$\hat{Q}^{2D}_{N\cdot M}(f) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \hat{Q}^{2D}_{n,m}(f).$$

Błąd kubatury złożonej będzie wówczas sumą błędów kubatur prostych. W przypadku, gdy dla każdego $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$ zachodzi $h_n = \Delta t$ oraz $w_m = \Delta s$, kubatura $\hat{Q}^{2D}_{N\cdot M}(f)$ przyjmuje postać kubatury równomiernej

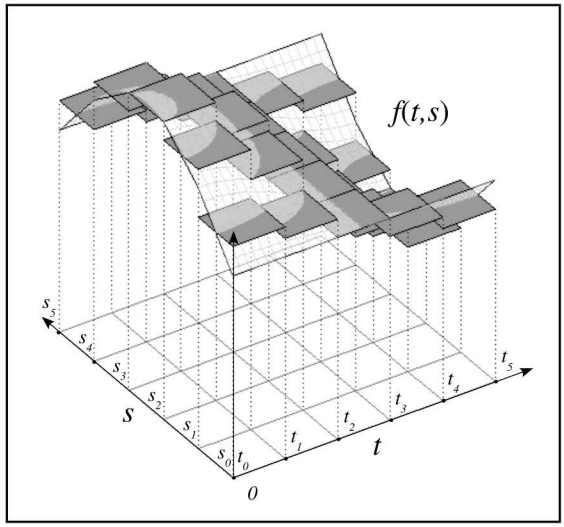
$$\hat{Q}^{2D}_{N\cdot M}(f) = \Delta t \Delta s \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f(\hat{t}_n, \hat{s}_m), \tag{3.23}$$

której błąd można wyznaczyć z poniższej zależności

$$\begin{aligned} \hat{\epsilon}^{2D}_{N\cdot M}(f) \;=\; & \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \left[\sum_{l=1}^{K/2-1} \left(\sum_{i=0}^{l} \frac{\Delta t^{2(l-i)+1} \Delta s^{2i+1}}{(2i+1)!(2(l-i)+1)!2^{2l}} \frac{\partial^{2l} f(\hat{t}_n, \hat{s}_m)}{\partial t^{2(l-i)} \partial s^{2i}} \right) + \right. \\ & \left. + \sum_{i=0}^{K/2} \frac{\Delta t^{K-2i+1} \Delta s^{2i+1}}{(2i+1)!(K-2i+1)!2^K} \frac{\partial^K f(\epsilon_n, \eta_m)}{\partial t^{K-2i} \partial s^{2i}} \right]. \end{aligned} \tag{3.24}$$

Wielkość tego błędu, podobnie jak (3.21), zależy wprost proporcjonalnie od wartości parzystych pochodnych cząstkowych funkcji $f(t, s)$ oraz parametrów Δt i Δs określających powierzchnie podprzedziałów, na których budowano kubatury $\hat{Q}^{2D}_{n,m}(f)$.

Na rysunku 3.9 zobrazowano sposób numerycznego obliczania wartości całki przy pomocy złożonej kubatury $\hat{Q}^{2D}_{N\cdot M}(f)$.



Rysunek 3.9: Przybliżone obliczanie całki $I^{2D}(f)$ przy pomocy złożonej kubatury $\hat{Q}^{2D}_{N\cdot M}(f)$ dla przypadku $N = M = 5$

Dwuwymiarowe dyskretne przekształcenie Fouriera

Zgodnie z argumentacją zawartą w rozdziale 2 wiadomo, iż dyskretne dwuwymiarowe przekształcenie Fouriera (2.19) umożliwia przybliżone obliczanie widma dwuwymiarowego całkowego przekształcenia Fouriera w punktach (ω_k, Ω_l), tzn. prawdziwa jest następująca zależność

$$X^{2D}(\omega_k, \Omega_l) \approx T_1 T_2 X^{2D}_{N\cdot M}(k, l)$$

dla $k = 0, \pm 1, ..., \pm\frac{N}{2}$ i $l = 0, \pm 1, ..., \pm\frac{M}{2}$. Wówczas korzystając z definicji dwuwymiarowego przekształcenia Fouriera (patrz wzór (2.19)), prawą stronę powyższej zależności można zapisać w postaci

$$T_1 T_2 X^{2D}_{N\cdot M}(k, l) = \Delta t \Delta s \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(t_n, s_m) e^{-i\omega_k t_n} e^{-i\Omega_l s_m},$$

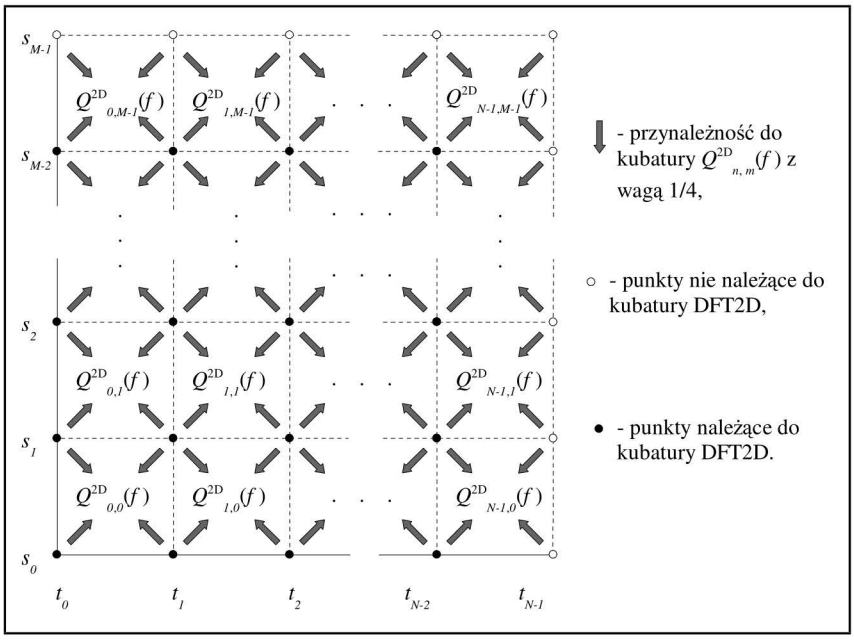
gdzie $\Delta t = T_1/N$ oraz $\Delta s = T_2/M$. Wyrażenie to dla każdej dwójki (k, l) jest kubaturą numerycznego obliczania całki $I^{2D}(f)$ z funkcji $f(t, s, k, l) = x(t, s)e^{-i\omega_k t}e^{-i\Omega_l s}$, po obszarze $[0, T_1] \times [0, T_2]$. Ponieważ funkcja $f(t, s, k, l)$ może przyjmować wartości zespolone, to całkę z tej funkcji należy rozpatrywać jako parę całek z jej części rzeczywistej i urojonej. Tym samym, dla każdej dwójki (k, l) powyższą kubaturę należy rozumieć jako parę dwóch kubatur numerycznych, które operują odpowiednio na składowej rzeczywistej i urojonej funkcji podcałkowej $f(t, s, k, l)$. Składowe te przyjmują następujące postaci

$$Re\{f(t, s, k, l)\} = x(t, s)\left[cos(\omega_k t)cos(\Omega_l s) - sin(\omega_k t)sin(\Omega_l s)\right],$$

$$Im\{f(t, s, k, l)\} = -ix(t, s)\left[sin(\omega_k t)cos(\Omega_l s) + cos(\omega_k t)sin(\Omega_l s)\right].$$

Zatem dyskretne dwuwymiarowe przekształcenie Fouriera, z dokładnością do stałych T_1T_2, stanowi zespół kubatur numerycznego obliczania całkowego przekształcenia Fouriera. Łatwo wykazać, iż kubatury te przybliżają wartość całek $X^{2D}(\omega_k, \Omega_l)$ objętościami prostopadłościanów o podstawach długości Δt i Δs. O ile punkty węzłowe wspomnianych kubatur leżą w punktach (t_n, s_m), gdzie $t_n = n\Delta t$ i $s_m = m\Delta s$, a nie w środkach obszarów $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$, to błąd przybliżenia wartości całki $X^{2D}(\omega_k, \Omega_l)$ nie może być wyrażony wzorem (3.24). Jednakże kubatury $T_1T_2X^{2D}_{N\cdot M}(k, l)$ można w łatwy sposób przekształcić do postaci kubatur $Q^{2D}_{N\cdot M}(f)$, poprzez wprowadzenie odpowiednich modyfikacji do ich współczynników węzłowych $f(t_n, s_m, k, l)$, tak jak to miało miejsce w przypadku jednowymiarowym.

Poniższy rysunek pozwala określić sposób modyfikacji współczynników węzłowych, który jest wymagany dla przekształcenia kubatur DFT2D do rozpatrywanej postaci (3.20). Symbolem strzałki $\uparrow$ oznaczono przynależność danego węzła do kubatury prostej $Q^{2D}_{n,m}(f)$ z wagą 1/4, natomiast symbole $\bullet$ oraz $\circ$ oznaczają odpowiednio węzły wchodzące w skład kubatury DFT2D, oraz te węzły, które nie występują w kubaturze DFT2D, ale wymagane są dla kubatury $Q^{2D}_{N\cdot M}(f)$. Wówczas porównując sposób doboru współczynników węzłowych dla kubatury $Q^{2D}_{N\cdot M}(f)$, z regułą doboru punktów węzłowych wynikającą z definicji dyskretnego dwuwymiarowego przekształcenia Fouriera, można wyznaczyć następujące modyfikacje współczynników węzłowych $f(t_n, s_m, k, l)$, które pozwalają na przekształcenie kubatury Fouriera do kubatury określonej zależnością



Rysunek 3.10: Dobór współczynników węzłowych dla kubatury $Q^{2D}_{N\cdot M}(f)$

(3.20). Ponieważ dla każdej pary (k,l) iloczyny funkcji wykładniczych $e^{-i\omega_k t_n}e^{-i\Omega_l s_m}$ są równe jedności przy parametrach n i m przyjmujących wartość 0 lub N (dla przypadku t_n) i M (dla przypadku s_m), to szukane modyfikacje współczynników węzłowych sprowadzają się do modyfikacji wartości próbek sygnału $x(t,s)$, na których operuje dyskretne dwuwymiarowe przekształcenie Fouriera. Szukane modyfikacje współczynników węzłowych określają wówczas poniższe zależności

$$x(t_0,s_0) = \frac{x(t_0,s_0)+x(t_N,s_0)+x(t_0,s_M)+x(t_N,s_M)}{4},$$
$$x(t_n,s_0) = \frac{x(t_n,s_0)+x(t_n,s_M)}{2} \tag{3.25}$$

dla $n = 1,2,...,N-1$ oraz

$$x(t_0,s_m) = \frac{x(t_0,s_m)+x(t_N,s_m)}{2}$$

dla $m = 1,2,...,M-1$. Uwzględniając te zmiany w zbiorze próbek sygnału, otrzymujemy dla każdej pary (k,l) kubaturę postaci (3.20), której błąd zależy od parzystych pochodnych cząstkowych funkcji $f(t,s,k,l)$.

Przyglądając się dokładnie rzeczywistej oraz urojonej części funkcji podcałkowej $f(t,s,k,l)$ zauważymy, iż w danym przypadku modyfikacja próbek, potrzebna dla uzyskania kubatury opisanej wzorem (3.20), jest wymagana zarówno dla kubatur związanych z częścią rzeczywistą, jak i urojoną tej funkcji. Jednakże w przypadku kubatury liczonej dla części urojonej wystarcza wyłącznie modyfikacja próbek $x(t_n,s_0)$ oraz

$x(t_0, s_m)$ dla $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$, co wynika z faktu, iż w punktach krańcowych $x(t_0, s_0)$, $x(t_N, s_0)$, $x(t_0, s_M)$ oraz $x(t_N, s_M)$ funkcja $Im\{f(t, s, k, l)\}$ jest zawsze równa zeru.

Dwuwymiarowe dyskretne przekształcenia kosinusowe i sinusowe drugiego oraz czwartego rodzaju

Dyskretne dwuwymiarowe przekształcenia kosinusowe i sinusowe drugiego oraz czwartego rodzaju (2.20) - (2.23) operują na dyskretnych próbkach sygnału $x(t, s)$ pobieranych w środkach przedziałów $[t_n, t_{n+1}] \times [s_m, s_{m+1}]$ dla $n = 0, 1, ..., N-1$ oraz $m = 0, 1, ..., M-1$. Zatem na podstawie wcześniejszej dyskusji można jednoznacznie stwierdzić, iż przekształcenia te, z dokładnością do stałych $T_1T_2/(p_kp_l)$ lub T_1T_2 (w zależności od rodzaju przekształcenia), stanowią dla każdej pary parametrów (k, l) złożone kubatury postaci (3.23). Błąd tych kubatur zależy od wartości parzystych pochodnych funkcji podcałkowych $f(t, s, k, l)$, gdzie funkcje te są iloczynami transformowanego sygnału $x(t, s)$ z odpowiednimi funkcjami bazowymi przekształcenia całkowego.

3.3. Algorytmy adaptacyjnego obliczania przekształceń jednowymiarowych

W pracach [7, 39] zaprezentowany został algorytm adaptacyjnego obliczania całek z wykorzystaniem kwadratur całkowania numerycznego. W algorytmie tym adaptacji podlega liczba punktów węzłowych kwadratury, które przypadają na przedział całkowania. Jako kryterium doboru liczby punktów przyjmuje się tutaj błąd numerycznego obliczania całki, którego wielkość przybliżana jest na podstawie wartości dwóch kwadratur o liczbie N i $2N$ punktów węzłowych.

W pierwszej części niniejszego rozdziału wykazano, iż dyskretne przekształcenia trygonometryczne stanowią kwadratury numerycznego obliczania przekształceń trygonometrycznych w postaci całkowej. Zatem do wyznaczania wartości przekształceń całkowych możliwe jest zastosowanie wskazanego podejścia adaptacyjnego, które bazować będzie na kwadraturach w postaci przekształceń dyskretnych. Tutaj adaptacji podlega liczba próbek sygnału wejściowego, którą dobiera się według kryterium: *dokładność obliczania reprezentacji sygnału w bazie danego przekształcenia całkowego - czas realizacji obliczeń.* Na wstępie przedstawiona zostanie jednak idea działania algorytmu adaptacyjnego opisanego w pracach [7, 39].

Wspomniany algorytm w sposób automatyczny określa rozmiary podprzedziałów stanowiących podział obszaru całkowania funkcji w taki sposób, ażeby przybliżenie wartości całki przy pomocy kwadratury numerycznej spełniało założone kryterium dokładności. W rozważanych przypadkach zakładamy stałą długość podprzedziałów, tzn. punkty węzłowe kwadratury rozłożone są na przedziale całkowania w sposób równomierny. Wówczas na kolejnych etapach adaptacji podprzedziały te dzielone są na połowę, a liczba punktów węzłowych N jest podwajana. Następnie przy użyciu reguły heurystycznej, oraz na podstawie wartości kwadratur N i $2N$-punktowych, w sposób przybliżony wyznacza się wartość rzeczywistego błędu obliczania całki poprzez kwadraturę numeryczną. Algorytm kończy działanie z chwilą, gdy przybliżona wartość błędu jest mniejsza lub równa założonej wartości dopuszczalnej ϵ. W ramach omówienia algorytmu adaptacyjnego, jako pierwszy rozpatrzony zostanie przypadek przekształceń jednowymiarowych, a w ich obrębie złożone kwadratury trapezów i prostokątów.

Postać złożonej kwadratury trapezów $Q_N(f)$ dla numerycznego obliczania wartości całki $I(f)$ z funkcji $f(t)$, została opisana wzorem (3.8). Z kolei błąd numerycznego obliczania wartości całki $I(f)$ przy pomocy kwadratury $Q_N(f)$, definiuje zależność (3.9). Wprowadzając dla uproszczenia zapisu następujące wyrażenie $R_I(f,n)$ postaci

$$R_I(f,n) = \sum_{l=2}^{K/2-1} \frac{f^{(2l)}(\hat{t}_n)2l}{2^{2l}(2l+1)!}\Delta t^{2l+1} + \frac{f^{(K)}(\xi_n)K}{2^K(K+1)!}\Delta t^{K+1},$$

to zależność (3.9) można zapisać jako

$$\epsilon_N(f) \;=\; I(f) - Q_N(f) = -\sum_{n=0}^{N-1}\left(\frac{f^{(2)}(\hat{t}_n)}{12}\Delta t^3 + R_I(f,n)\right), \tag{3.26}$$

gdzie $\hat{t}_n = (2n+1)\frac{\Delta t}{2}$ dla $n = 0, 1, ..., N-1$ jest punktem środkowym dla podprzedziału $[t_n, t_{n+1}]$ o długości Δt. Podobnie wzór pozwalający na obliczenie wartości błędu dla kwadratury $Q_{2N}(f)$, opartej o $2N$ punkty węzłowe, przyjmie postać wyrażenia

$$\epsilon_{2N}(f) \;=\; I(f) - Q_{2N}(f) = -\sum_{n=0}^{2N-1}\left(\frac{f^{(2)}(\tilde{t}_n)}{12}\left(\frac{\Delta t}{2}\right)^3 + R_{II}(f,n)\right)$$

dla $\tilde{t}_n = (2n+1)\Delta t/4$, przy $n = 0, 1, ..., 2N-1$, gdzie

$$R_{II}(f,n) = \sum_{l=2}^{K/2-1} \frac{f^{(2l)}(\tilde{t}_n)2l}{2^{2l}(2l+1)!}\left(\frac{\Delta t}{2}\right)^{2l+1} + \frac{f^{(K)}(\xi_n)K}{2^K(K+1)!}\left(\frac{\Delta t}{2}\right)^{K+1},$$

oraz punkt $\tilde{t}_n$ oznacza środek podprzedziału o długości $\frac{\Delta t}{2}$. Powyższą zależność można dodatkowo rozpisać względem parzystych i nieparzystych indeksów n jako sumę

$$\begin{aligned}\epsilon_{2N}(f) \;&=\; -\sum_{n=0}^{N-1}\left(\frac{f^{(2)}(\tilde{t}_{2n})}{12}\left(\frac{\Delta t}{2}\right)^3 + \frac{f^{(2)}(\tilde{t}_{2n+1})}{12}\left(\frac{\Delta t}{2}\right)^3 + \right. \\ &+\; \left. R_{II}(f,2n) + R_{II}(f,2n+1)\right).\end{aligned} \tag{3.27}$$

Wówczas uzależniając wartości drugiej pochodnej w punktach $\hat{t}_n$, od jej wartości w punktach $\tilde{t}_{2n}$ i $\tilde{t}_{2n+1}$ w sposób następujący: $f^{(2)}(\hat{t}_n) = f^{(2)}(\tilde{t}_{2n}) + r_I(\hat{t}_n)$ oraz $f^{(2)}(\hat{t}_n) = f^{(2)}(\tilde{t}_{2n+1}) + r_{II}(\hat{t}_n)$ dla $n = 0, 1, ..., N-1$, to wyrażenie (3.27) można zapisać ostatecznie w postaci wzoru

$$\begin{aligned}\epsilon_{2N}(f) \;&=\; -\sum_{n=0}^{N-1}\left(\frac{f^{(2)}(\hat{t}_{2n})}{4\cdot 12}\Delta t^3 - \frac{(r_I(\hat{t}_n) + r_{II}(\hat{t}_n))}{8\cdot 12}\Delta t^3 + \right. \\ &+\; \left. R_{II}(f,2n) + R_{II}(f,2n+1)\right),\end{aligned}$$

który po porównaniu z wyrażeniem (3.26) daje zależność łączącą oba błędy $\epsilon_N(f)$ oraz $\epsilon_{2N}(f)$ w następujący sposób

$$\epsilon_{2N}(f) = \frac{1}{4}\epsilon_N(f) + R_{III}(f), \tag{3.28}$$

gdzie

$$R_{III}(f) = \sum_{n=0}^{N-1} \left(\frac{(r_I(\hat{t}_n) + r_{II}(\hat{t}_n))}{8 \cdot 12} \Delta t^3 + \right.$$
$$\left. + \frac{R_I(f,n)}{4} - R_{II}(f,2n) - R_{II}(f,2n+1) \right).$$

Po podstawieniu za $\epsilon_N(f) = I(f) - Q_N(f)$ oraz za $\epsilon_{2N}(f) = I(f) - Q_{2N}(f)$, powyższy wzór można przekształcić w sposób pozwalający na obliczanie wartości błędu $\epsilon_N(f)$ lub $\epsilon_{2N}(f)$ na podstawie różnicy wartości kwadratur $Q_N(f)$ i $Q_{2N}(f)$. Rozważmy drugi przypadek dotyczący wartości błędu $\epsilon_{2N}(f)$. Wówczas w wyniku odpowiednich przekształceń wzoru (3.28) otrzymujemy kolejno

$$I(f) - Q_{2N}(f) = \tfrac{1}{4}(I(f) - Q_N(f)) + R_{III}(f),$$

$$\frac{3}{4}(I(f) - Q_{2N}(f)) + \frac{1}{4}(I(f) - Q_{2N}(f)) = \frac{1}{4}(I(f) - Q_N(f)) + R_{III}(f),$$

$$3(I(f) - Q_{2N}(f)) = (Q_{2N}(f) - Q_N(f)) + 4R_{III}(f),$$

$$I(f) - Q_{2N}(f) = \frac{1}{3}(Q_{2N}(f) - Q_N(f)) + \frac{4}{3}R_{III}(f).$$

Rezultat powyższych przekształceń zapisujemy w postaci wzoru

$$\epsilon_{2N}(f) = \frac{1}{3}(Q_{2N}(f) - Q_N(f)) + \frac{4}{3}R_{III}(f). \tag{3.29}$$

Wyrażenie to posiada skomplikowaną i w praktyce nie możliwą do wykorzystania postać. Wartość członu $R_{III}(f)$ jest uzależniona od pochodnych wyższych rzędów funkcji podcałkowej, których wartości w praktyce nie są znane. Stąd w celu uzyskania odpowiedniej dla praktycznych zadań cyfrowego przetwarzania danych formy wyrażenia (3.29), wymagane jest wprowadzenie pewnych uproszczeń.

Po pierwsze przyjmujemy, że składowa błędu związana z wyższymi niż druga pochodnymi funkcji podcałkowej jest pomijalnie mała, tzn.

$$\sum_{n=0}^{N-1} \left(\frac{R_I(f,n)}{4} - R_{II}(f,2n) - R_{II}(f,2n+1) \right) \approx 0.$$

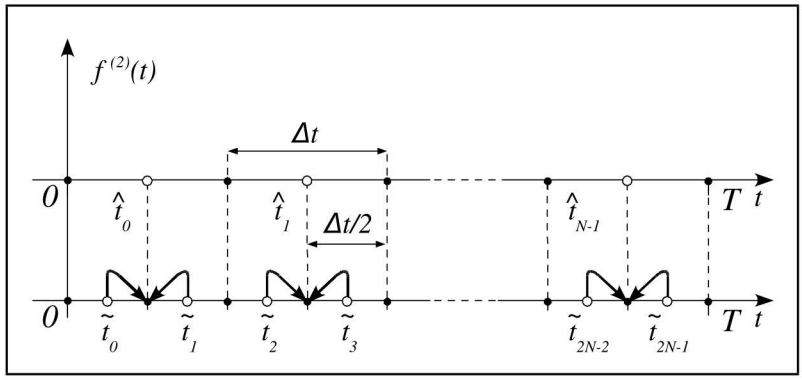
Takie założenie można przyjąć w chwili, gdy pochodne wyższych rzędów są ograniczone i po przeskalowaniu przez czynnik wynikający z konstrukcji wyrażenia na błąd, przyjmują wartości bliskie zeru. Zgodnie z wzorem (3.9) przyczynek do błędu związany z $2l$-tą pochodną zależy od przeskalowanej wartości kroku dyskretyzacji podniesionego do $2l+1$ potęgi, tzn. od czynnika postaci $2l/(2^{2l}(2l+1)!) \cdot (T/N)^{2l+1}$. Dla liczby $N = 128$ próbek i $T = 1$ czynnik ten już dla 4-tej pochodnej przyjmuje wartość rzędu 10^{-14}, podczas, gdy dla drugiej pochodnej osiąga wartość rzędu 10^{-8}.

Drugie założenie dotyczy wartości drugiej pochodnej funkcji podcałkowej. W tym przypadku przyjmuje się, że wartości te w punktach $\tilde{t}_{2n}$ oraz $\tilde{t}_{2n+1}$ są bliskie wartości drugiej pochodnej w punkcie $\hat{t}_n$ dla $n = 0, 1, ..., N-1$. Założenie takie będzie prawdziwe

w chwili, gdy druga pochodna funkcji podcałkowej $f(t)$ wykazywać będzie małe zmiany wartości na przedziałach $[\tilde{t}_{2n}, \tilde{t}_{2n+1}]$ o długości $\frac{\Delta t}{2}$, gdzie $n = 0, 1, ..., N-1$. Wówczas zachodzić będzie następująca zależność

$$\underset{n=0,1,...,N-1}{\forall} r_I(\hat{t}_n) \approx 0, \text{ oraz } \underset{n=0,1,...,N-1}{\forall} r_{II}(\hat{t}_n) \approx 0.$$

Na rysunku (3.11) pokazano punkty, w których w świetle przyjętego założenia, wartości drugiej pochodnej są w przybliżeniu sobie równe. Przy pomocy strzałki oznaczono "przybliżoną równość" wartości drugiej pochodnej.



Rysunek 3.11: Punkty na osi czasu, w których zakłada się, iż druga pochodna $f^{(2)}(t)$ przyjmuje w przybliżeniu takie same wartości. Strzałką oznaczono przybliżona równość wartości drugiej pochodnej funkcji podcałkowej

Przyjęcie powyższych założeń sprawia, iż przybliżoną wartość błędu $\epsilon_{2N}(f)$ można wyznaczyć zgodnie ze wzorem (3.29) przy pomocy następującej zależności

$$\epsilon_{2N}(f) \approx \frac{1}{3}(Q_{2N}(f) - Q_N(f)).$$

Wówczas do jej obliczenia wystarcza znajomość wartości kwadratur $Q_{2N}(f)$ i $Q_N(f)$, które operują odpowiednio na punktach węzłowych $t_n = n\frac{\Delta t}{2}$ dla $n = 0, 1, ..., 2N-1$ oraz $t_n = n\Delta t$ dla $n = 0, 1, ..., N-1$, gdzie $\Delta t = T/N$. Przy założeniu, że znak błędu nie ma znaczenia, jego przybliżoną wartość oznaczoną symbolem $\epsilon^O_{2N}(f)$, obliczamy jako moduł z różnicy wartości kwadratur podzielony przez czynnik 3, tzn. $\epsilon^O_{2N}(f) = |Q_{2N}(f) - Q_N(f)|/3$. Ostatecznie jako wyrażenie kontrolujące dobór liczby $2N$ punktów węzłowych kwadratury $Q_{2N}(f)$, w ramach algorytmu adaptacyjnego, przyjmuje się nierówność postaci

$$\epsilon^O_{2N}(f) = \frac{1}{3}|Q_{2N}(f) - Q_N(f)| \leqslant \epsilon, \tag{3.30}$$

gdzie ϵ to dopuszczalna wartość błędu numerycznego obliczania całki $I(f)$ przy pomocy kwadratury $Q_{2N}(f)$. Nierówność ta, ze względu na przyjęte założenia, ma charakter heurystyczny. Jej skuteczność została empirycznie zweryfikowana w rozdziale 7.

W analogiczny sposób otrzymujemy wyrażenie kontrolujące dobór próbek dla złożonej kwadratury prostokątów określonej wzorem (3.10). Ponieważ także i w tym przypadku błąd kwadratury zależy od drugiej pochodnej funkcji podcałkowej (patrz wzór

(3.11)), to wyrażenie to przyjmie identyczną postać, tak jak w przypadku kwadratury trapezów, tj. postać opisaną nierównością (3.30).

W pierwszej sekcji niniejszego rozdziału wykazano, iż jednowymiarowe dyskretne przekształcenie Fouriera, oraz kosinusowe i sinusowe przekształcenia drugiego i czwartego rodzaju, stanowią odpowiednio zespoły kwadratur trapezów lub prostokątów dla numerycznego obliczania całkowych przekształceń Fouriera. Kwadratury te są sparametryzowane indeksem k określającym pulsacje ω_k, dla których w sposób przybliżony obliczane są wartości przekształceń całkowych. Jako pierwszy rozważmy przypadek dyskretnego przekształcenia Fouriera.

Wiadomo, że przekształcenie to po odpowiedniej modyfikacji próbki o indeksie zerowym, stanowi zespół złożonych kwadratur trapezów. Ponieważ funkcje podcałkowe $f(t,k)$ przekształcenia Fouriera są zespolone, to wartość całki Fouriera $X(\omega_k)$, oraz wartości kwadratur: $2N$-punktowej $X_{2N}(k)$ i N-punktowej $X_N(k)$, a także wartość błędu $\epsilon_{2N}(k)$ dla kwadratury $2N$-punktowej są liczbami zespolonymi. Skoro kwadratury związane z częściami rzeczywistą i urojoną funkcji podcałkowej są kwadraturami trapezów, to ich błąd zależy od parzystych wartości pochodnych funkcji podcałkowej. Stąd możliwe jest zastosowanie przedstawionego wyrażenia kontroli doboru liczby próbek również dla przypadku przekształcenia Fouriera, tj. osobno dla części rzeczywistej i urojonej kwadratury $X_N(k)$. Wówczas moduł z różnicy wartości kwadratur rozumieć będziemy w sensie modułu liczby zespolonej $\|z\| = (Re\{z\}^2 + Im\{z\}^2)^{1/2}$. O ile część rzeczywista wyrażenia na błąd $Re\{\epsilon_{2N}(k)\} \approx TRe\{X_{2N}(k) - X_N(k)\}/3$, a także jego część urojona $Im\{\epsilon_{2N}(k)\} \approx TIm\{X_{2N}(k) - X_N(k)\}/3$, to zachodzić będzie również związek postaci $\|\epsilon_{2N}(k)\| \approx T\|X_{2N}(k) - X_N(k)\|/3$. Zatem wyrażenie kontrolujące dobór liczby próbek dla każdej wartości k, przyjmie w przypadku przekształcenia Fouriera postać

$$\epsilon^O_{2N}(k) = \frac{T}{3}\|X_{2N}(k) - X_N(k)\| \leqslant \epsilon(k).$$

Nierówność ta pozwala jedynie na kontrolę liczby próbek dla poszczególnych kwadratur związanych z kolejnymi wartościami k. Stąd w pracy [52] zaproponowano wyrażenie oceny błędu całościowego, które skonstruowano w silnej metryce Czebyszewa (metryce maksymalnego odchylenia, z ang. *maximum difference* (MD) [124])

$$\epsilon^O_{MD} = \max_{k=0,1,...,\frac{N}{2}} \left\{ \frac{1}{3}\|X_{2N}(k) - X_N(k)\| \right\} \leqslant \epsilon, \tag{3.31}$$

gdzie ϵ to dopuszczalna w danej metryce wartość błędu. Wyrażenie to umożliwia dobór liczby $2N$ próbek sygnału, odpowiedniej dla wszystkich kwadratur $X_{2N}(k)$. Zakres stosowania indeksu k wynika z własności symetrii widma amplitudowego dyskretnego przekształcenia Fouriera (patrz Własność 2.3.1) postaci $\|X_N(k)\| = \|X_N(N-k)\|$ dla $k = 0, 1, ..., N/2$, która jest prawdziwa w chwili, gdy transformowany sygnał $x(t)$ jest sygnałem przyjmującym wyłącznie wartości rzeczywiste. Ponadto bez utraty ogólności rozważań (patrz rozdział 2) w powyższym wzorze przyjęto $T = 1$. Z wykorzystaniem wyrażenia (3.31) możliwa jest wówczas budowa algorytmu adaptacyjnego obliczania przekształcenia Fouriera (ADFT).

ALGORYTM 3.1 *(Algorytm adaptacyjnego obliczania przekształcenia Fouriera)*
Na wejściu dany musi być sygnał $x(t)$ o wartościach rzeczywistych, określony na przedziale $[0,1]$, zbiór jego N próbek pobranych w punktach $t_n = n\Delta t$, gdzie $\Delta t = 1/N$,

a także liczba $M \leqslant N$ (przyjmujemy M parzyste) współczynników widmowych branych pod uwagę w procesie adaptacji oraz dopuszczalna wartość błędu ϵ.

1. *Oblicz $X_N(k)$ dla $k = 0, 1, ..., \frac{M}{2}$.*
2. *Dobierz N próbek w punktach $t_{2n+1} = (2n+1)\Delta t/2$ dla $n = 0, 1, ..., N-1$.*
3. *Na podstawie zbioru $2N$ próbek oblicz $X_{2N}(k)$ dla $k = 0, 1, ..., \frac{M}{2}$, oraz wartości $X_{2N}(2N-k)$ dla $k = 1, 2, ..., \frac{M}{2} - 1$.*
4. *Oblicz przybliżoną wartość ϵ_{MD}^{O} na podstawie następującej zależności $\epsilon_{MD}^{O} = \max\limits_{k=0,1,...,\frac{M}{2}} \left\{ \frac{1}{3} \| X_{2N}(k) - X_N(k) \| \right\}$. Jeżeli $\epsilon_{MD}^{O} \leqslant \epsilon$ to skocz do 6.*
5. *Podstaw $X_N(k) = X_{2N}(k)$ dla $k = 0, 1, ..., \frac{M}{2}$, $N = N \cdot 2$ oraz $\Delta t = \Delta t/2$. Skocz do 2.*
6. *Wypisz $X_{2N}(k)$ dla $k = 0, 1, ..., \frac{M}{2}$, $X_{2N}(2N-k)$ dla $k = 1, 2, ..., \frac{M}{2} - 1$, oraz wartości $2N$ i ϵ_{MD}^{O}.*

Koniec.

Przedstawiony algorytm (3.1) pozwala na dobór liczby $2N$ próbek, która jest wystarczająca do obliczenia M niskoczęstotliwościowych współczynników widmowych $X_{2N}(k)$, z zadanym w metryce Czebyszewa dopuszczalnym błędem ϵ.

Dla kosinusowego i sinusowego przekształcenia Fouriera funkcje bazowe przyjmują wartości rzeczywiste, a dyskretne przekształcenia drugiego i czwartego rodzaju można rozważać jako złożone kwadratury prostokątów, również sparametryzowane k. Wówczas wyrażenie kontrolujące dobór liczby $2N$ próbek w metryce Czebyszewa przyjmie postać

$$\epsilon_{MD}^{OPII} = \max_{k=0,1,...,N-1} \left\{ \frac{1}{3p_k} \left| X_{2N}^{PII}(k) - X_N^{PII}(k) \right| \right\} \leqslant \epsilon \tag{3.32}$$

dla przekształceń drugiego rodzaju, oraz postać

$$\epsilon_{MD}^{OPIV} = \max_{k=0,1,...,N-1} \left\{ \frac{1}{3} \left| X_{2N}^{PIV}(k) - X_N^{PIV}(k) \right| \right\} \leqslant \epsilon \tag{3.33}$$

dla przypadku przekształceń rodzaju czwartego. W powyższych wzorach P symbolizuje przekształcenie kosinusowe lub sinusowe (C - kosinusowe, S - sinusowe), natomiast ϵ to dopuszczalna w danej metryce wartość błędu. Również i w tym przypadku przyjęto $T = 1$. Ponieważ przekształcenia te nie wykazują symetrii takiej jak dyskretne przekształcenie Fouriera, to zakres stosowania parametru k jest w danym przypadku następujący $k = 0, 1, ..., N-1$.

Poniżej zamieszczono algorytm adaptacyjnego obliczania całkowych kosinusowych i sinusowych przekształceń Fouriera z doborem liczby próbek, która jest kontrolowana w zależności od rodzaju dyskretnego przekształcenia poprzez jedno z wprowadzonych wcześniej wyrażeń (3.32) lub (3.33). Algorytm taki, w zależności od rodzaju przekształcenia dyskretnego, oznaczać będziemy jako ADCT-II (ADCT-IV) dla przypadku przekształceń kosinusowych, oraz ADST-II (ADST-IV) dla przekształceń sinusowych.

ALGORYTM 3.2 *(Algorytm adaptacyjnego obliczania kosinusowych i sinusowych przekształceń Fouriera)*
Na wejściu dany musi być sygnał $x(t)$ określony na przedziale $[0,1]$, zbiór jego N próbek pobranych w punktach $t'_n = (n+1/2)\Delta t$, gdzie $\Delta t = 1/N$, liczba $M \leqslant N$ współczynników widmowych, które brane są pod uwagę w procesie adaptacji, typ przekształcenia P, oraz dopuszczalna wartość błędu ϵ.

1. *Oblicz $X_N^{PII}(k)$ $(X_N^{PIV}(k))$ dla $k = 0, 1, ..., M-1$.*
2. *Pobierz $2N$ próbek w punktach $t'_n = (n+1/2)\Delta t/2$ dla $n = 0, 1, ..., 2N-1$.*
3. *Na podstawie zbioru $2N$ próbek oblicz $X_{2N}^{PII}(k)$ $(X_{2N}^{PIV}(k))$ dla $k = 0, 1, ..., M-1$.*
4. *Oblicz przybliżoną wartość ϵ_{MD}^{OPII} (ϵ_{MD}^{OPIV}) na podstawie następującej zależności*

$$\epsilon_{MD}^{OPII} = \max_{k=0,1,...,M-1} \left\{ \frac{1}{3p_k} \left| X_{2N}^{PII}(k) - X_N^{PII}(k) \right| \right\}$$

$$(\epsilon_{MD}^{OPIV} = \max_{k=0,1,...,M-1} \left\{ \frac{1}{3} \left| X_{2N}^{PIV}(k) - X_N^{PIV}(k) \right| \right\}).$$

   Jeżeli ϵ_{MD}^{OPII} $(\epsilon_{MD}^{OPIV}) \leqslant \epsilon$ to skocz do 6.

5. *Podstaw $X_N^{PII}(k) = X_{2N}^{PII}(k)$ $(X_N^{PIV}(k) = X_{2N}^{PIV}(k))$ dla $k = 0, 1, ..., M-1$, $N = N \cdot 2$ oraz $\Delta t = \Delta t/2$. Skocz do 2.*
6. *Wypisz $X_{2N}^{PII}(k)$ $(X_{2N}^{PII}(k))$ dla $k = 0, 1, ..., M-1$, oraz wartości $2N$ i ϵ_{MD}^{OPII} (ϵ_{MD}^{OPIV}).*

Powyższy algorytm pozwala na dobór liczby $2N$ próbek, która jest wystarczająca do obliczenia M niskoczęstotliwościowych współczynników widmowych $X_{2N}^{PII}(k)$ $(X_{2N}^{PIV}(k))$ z zadanym w metryce Czebyszewa dopuszczalnym błędem ϵ.

3.4. Algorytmy adaptacyjnego obliczania przekształceń dwuwymiarowych

Przedstawiony w pracach [7] i [39] algorytm adaptacyjnego obliczania całek można z powodzeniem przenieść na przypadek funkcji dwóch zmiennych $f(t,s)$ oraz kubatur całkowania numerycznego, które omówiono w sekcji 3.2. W tym celu należy wprowadzić odpowiednie wyrażenie pozwalające na kontrolę doboru liczby punktów węzłowych kubatury. W dalszej części niniejszej sekcji wyrażenie takie opracowane zostanie na przykładzie złożonej kubatury trapezów o równomiernie rozłożonych punktach węzłowych, której postać definiuje zależność (3.20). Błąd numerycznego obliczania wartości całki przy pomocy wspomnianej kubatury można wówczas wyrazić jako (patrz 3.21)

$$\epsilon_{N\cdot M}^{2D}(f) = -\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left[\sum_{l=1}^{K/2-1}\left(\sum_{i=0}^{l}\frac{(2l(2i+1)-4i^2)\Delta t^{2(l-i)+1}\Delta s^{2i+1}}{(2i+1)!(2(l-i)+1)!2^{2l}}\frac{\partial^{2l} f(\hat{t}_n,\hat{s}_m)}{\partial t^{2(l-i)}\partial s^{2i}}\right) + \right.$$
$$\left. + \sum_{i=0}^{K/2}\frac{(K(2i+1)-4i^2)\Delta t^{K-2i+1}\Delta s^{2i+1}}{(2i+1)!(K-2i+1)!2^K}\frac{\partial^K f(\epsilon_n,\eta_m)}{\partial t^{K-2i}\partial s^{2i}}\right].$$

Wyrażenie to można rozpisać jako sumę przyczynków do błędu związanych z drugimi pochodnymi cząstkowymi funkcji podcałkowej oraz z pochodnymi wyższych rzędów, otrzymując w rezultacie następującą postać

$$\begin{aligned}\epsilon^{2D}_{N\cdot M}(f) &= -\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left[\frac{\Delta t^3\Delta s}{12}\frac{\partial^2 f(\hat{t}_n,\hat{s}_m)}{\partial t^2}+\frac{\Delta t\Delta s^3}{12}\frac{\partial^2 f(\hat{t}_n,\hat{s}_m)}{\partial s^2}+\right.\\ &= \sum_{l=2}^{K/2-1}\left(\sum_{i=0}^{l}\frac{(2l(2i+1)-4i^2)\Delta t^{2(l-i)+1}\Delta s^{2i+1}}{(2i+1)!(2(l-i)+1)!2^{2l}}\frac{\partial^{2l} f(\hat{t}_n,\hat{s}_m)}{\partial t^{2(l-i)}\partial s^{2i}}\right)+\\ &+ \left.\sum_{i=0}^{K/2}\frac{(K(2i+1)-4i^2)\Delta t^{K-2i+1}\Delta s^{2i+1}}{(2i+1)!(K-2i+1)!2^K}\frac{\partial^K f(\epsilon_n,\eta_m)}{\partial t^{K-2i}\partial s^{2i}}\right].\end{aligned}$$

Zakładając, podobnie jak w przypadku kwadratur dla funkcji jednej zmiennej, że przyczynek do błędu kubatury (3.20), związany z wyższymi niż drugie pochodne cząstkowe, jest pomijalnie mały i bliski zeru, to wartość błędu $\epsilon^{2D}_{N\cdot M}(f)$ można w sposób przybliżony wyznaczyć przy pomocy zależności

$$\epsilon^{2D}_{N\cdot M}(f) \approx -\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left[\frac{\Delta t^3\Delta s}{12}\frac{\partial^2 f(\hat{t}_n,\hat{s}_m)}{\partial t^2}+\frac{\Delta t\Delta s^3}{12}\frac{\partial^2 f(\hat{t}_n,\hat{s}_m)}{\partial s^2}\right].$$

Dla kubatury o liczbie $2N$ na $2M$ punktów węzłowych wyrażenie to przyjmie postać

$$\epsilon^{2D}_{2N\cdot 2M}(f) \approx -\sum_{n=0}^{2N-1}\sum_{m=0}^{2M-1}\left[\frac{\Delta t^3\Delta s}{12\cdot 16}\frac{\partial^2 f(\tilde{t}_n,\tilde{s}_m)}{\partial t^2}+\frac{\Delta t\Delta s^3}{12\cdot 16}\frac{\partial^2 f(\tilde{t}_n,\tilde{s}_m)}{\partial s^2}\right], \quad (3.34)$$

gdzie punkty węzłowe $\tilde{t}_n$ i $\tilde{s}_m$ wyznaczamy na podstawie $\tilde{t}_n = (2n+1)\Delta t/2$ oraz $\tilde{s}_m = (2m+1)\Delta s/2$ dla $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$, przy czym $\Delta t = T_1/N$ oraz $\Delta s = T_2/M$. Wzór (3.34) można rozpisać względem parzystych i nieparzystych indeksów n i m, uzyskując w rezultacie poniższą zależność

$$\begin{aligned}\epsilon^{2D}_{2N\cdot 2M}(f) &\approx -\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left[\frac{\Delta t^3\Delta s}{12\cdot 16}\left(\frac{\partial^2 f(\tilde{t}_{2n},\tilde{s}_{2m})}{\partial t^2}+\frac{\partial^2 f(\tilde{t}_{2n+1},\tilde{s}_{2m})}{\partial t^2}+\right.\right.\\ &+ \left.\frac{\partial^2 f(\tilde{t}_{2n},\tilde{s}_{2m+1})}{\partial t^2}+\frac{\partial^2 f(\tilde{t}_{2n+1},\tilde{s}_{2m+1})}{\partial t^2}\right)+\\ &+ \frac{\Delta t\Delta s^3}{12\cdot 16}\left(\frac{\partial^2 f(\tilde{t}_{2n},\tilde{s}_{2m})}{\partial s^2}+\frac{\partial^2 f(\tilde{t}_{2n+1},\tilde{s}_{2m})}{\partial s^2}+\right.\\ &+ \left.\left.\frac{\partial^2 f(\tilde{t}_{2n},\tilde{s}_{2m+1})}{\partial s^2}+\frac{\partial^2 f(\tilde{t}_{2n+1},\tilde{s}_{2m+1})}{\partial s^2}\right)\right].\end{aligned} \quad (3.35)$$

Ponadto przyjmując, iż drugie pochodne funkcji podcałkowej $f(t,s)$, liczone po zmiennych t i s, są w punktach $(\tilde{t}_{2n},\tilde{s}_{2m})$, $(\tilde{t}_{2n+1},\tilde{s}_{2m})$, $(\tilde{t}_{2n},\tilde{s}_{2m+1})$ oraz $(\tilde{t}_{2n+1},\tilde{s}_{2m+1})$ w przybliżeniu równe co do wartości tym pochodnym w punkcie $(\hat{t}_n,\hat{s}_m)$ dla $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$, tzn.

$$\underset{\substack{n=0,1,\ldots,N-1\\ m=0,1,\ldots,M-1}}{\forall} = \begin{cases} \dfrac{\partial^2 f(\tilde{t}_{2n}, \tilde{s}_{2m})}{\partial t^2} \approx \dfrac{\partial^2 f(\hat{t}_n, \hat{s}_m)}{\partial t^2}, \\ \dfrac{\partial^2 f(\tilde{t}_{2n+1}, \tilde{s}_{2m})}{\partial t^2} \approx \dfrac{\partial^2 f(\hat{t}_n, \hat{s}_m)}{\partial t^2}, \\ \dfrac{\partial^2 f(\tilde{t}_{2n}, \tilde{s}_{2m+1})}{\partial t^2} \approx \dfrac{\partial^2 f(\hat{t}_n, \hat{s}_m)}{\partial t^2}, \\ \dfrac{\partial^2 f(\tilde{t}_{2n+1}, \tilde{s}_{2m+1})}{\partial t^2} \approx \dfrac{\partial^2 f(\hat{t}_n, \hat{s}_m)}{\partial t^2}, \end{cases}$$

oraz

$$\underset{\substack{n=0,1,\ldots,N-1\\ m=0,1,\ldots,M-1}}{\forall} = \begin{cases} \dfrac{\partial^2 f(\tilde{t}_{2n}, \tilde{s}_{2m})}{\partial s^2} \approx \dfrac{\partial^2 f(\hat{t}_n, \hat{t}_m)}{\partial s^2}, \\ \dfrac{\partial^2 f(\tilde{t}_{2n+1}, \tilde{t}_{2m})}{\partial s^2} \approx \dfrac{\partial^2 f(\hat{t}_n, \hat{t}_m)}{\partial s^2}, \\ \dfrac{\partial^2 f(\tilde{t}_{2n}, \tilde{t}_{2m+1})}{\partial s^2} \approx \dfrac{\partial^2 f(\hat{t}_n, \hat{t}_m)}{\partial s^2}, \\ \dfrac{\partial^2 f(\tilde{t}_{2n+1}, \tilde{t}_{2m+1})}{\partial s^2} \approx \dfrac{\partial^2 f(\hat{t}_n, \hat{t}_m)}{\partial s^2}, \end{cases}$$

to wówczas wyrażenie (3.35) można zapisać w prostszej postaci, łączącej ze sobą wyrażenia na błędy $\epsilon^{2D}_{2N\cdot 2M}(f)$ oraz $\epsilon^{2D}_{N\cdot M}(f)$, jako

$$\epsilon^{2D}_{2N\cdot 2M}(f) \approx -\frac{1}{4}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left[\frac{\Delta t^3 \Delta s}{12}\frac{\partial^2 f(\hat{t}_n, \hat{s}_m)}{\partial t^2} + \frac{\Delta t \Delta s^3}{12}\frac{\partial^2 f(\hat{t}_n, \hat{s}_m)}{\partial s^2}\right].$$

Powyższe wyrażenie jest równoważne zależności

$$\epsilon^{2D}_{2N\cdot 2M}(f) \approx \frac{1}{4}\epsilon^{2D}_{N\cdot M}(f). \tag{3.36}$$

Następnie odpowiednio przekształcając wzór (3.36) otrzymujemy wyrażenie pozwalające na przybliżone obliczenie wartości błędu $\epsilon^{2D}_{2N\cdot 2M}(f)$ przy pomocy dwóch kubatur, operujących odpowiednio na N na M oraz $2N$ na $2M$ punktach węzłowych. Wyrażenie to przyjmuje następującą postać

$$\epsilon^{2D}_{2N\cdot 2M}(f) \approx \frac{1}{3}\left(Q^{2D}_{2N\cdot 2M}(f) - Q^{2D}_{N\cdot M}(f)\right).$$

Przyjmijmy dodatkowo, że znak błędu nie ma znaczenia, a przybliżoną wartość błędu kubatury $2N$ na $2M$ punktowej oznaczmy jako $\epsilon^{2DO}_{2N\cdot 2M}(f)$. Nierówność kontrolująca dobór liczby punktów węzłowych kubatury $Q^{2D}_{2N\cdot 2M}(f)$ przyjmie wówczas postać przeskalowanego modułu z różnicy wartości kubatur $Q^{2D}_{2N\cdot 2M}(f)$ oraz $Q^{2D}_{N\cdot M}(f)$

$$\epsilon^{2DO}_{2N\cdot 2M}(f) = \frac{1}{3}\left|Q^{2D}_{2N\cdot 2M}(f) - Q^{2D}_{N\cdot M}(f)\right| \leqslant \epsilon, \tag{3.37}$$

gdzie ϵ to dopuszczalna wartość błędu.

Ponieważ dla kubatury opisanej wzorem (3.23) błąd zależy również od wartości parzystych pochodnych cząstkowych funkcji podcałkowej, to wyrażenie kontrolujące dobór liczby $2N$ na $2M$ punktów węzłowych dla tej kubatury, przyjmie identyczną postać nierówności (3.37).

W dyskusji przeprowadzonej w sekcji 3.3 wykazano, iż dyskretne dwuwymiarowe przekształcenie Fouriera, a także dyskretne dwuwymiarowe przekształcenia kosinusowe i sinusowe drugiego i czwartego rodzaju, stanowią kubatury numeryczne dla całkowych przekształceń Fouriera. Zatem możliwe jest skonstruowanie dla wskazanych przekształceń algorytmów adaptacyjnych, które pozwolą na automatyczny dobór liczby $2N$ na $2M$ próbek transformowanego sygnału, zgodnie z kryterium: *dokładność reprezentacji sygnału w bazie przekształcenia całkowego - czas realizacji obliczeń.* Do kontroli liczby próbek wymagane są wyrażenia pozwalające na ocenę błędu całościowego, tzn. liczonego po zbiorze wartości k i l określających pulsacje ω_k oraz Ω_l, dla których obliczone mają być wartości przekształcenia całkowego. Przez analogię do przypadku jednowymiarowego zaproponowane wyrażenia skonstruowano w metryce Czebyszewa.

Dla dwuwymiarowego dyskretnego przekształcenia Fouriera wyrażenie oceny błędu całościowego przyjmie wówczas postać następującej nierówności

$$\epsilon_{MD}^{2DO}(k,l) = \max_{B}\left\{\frac{T_1T_2}{3}\|X_{2N\cdot 2M}^{2D}(k,l) - X_{N\cdot M}^{2D}(k',l')\|\right\} \leqslant \epsilon,$$

gdzie

$$B \triangleq \left\{(k,l,k',l'): \begin{array}{c} k=0,1,...,\frac{N_1}{2},l=0,1,...,\frac{M_1}{2},k'=k \text{ i } l'=l \\ k=2N-(\frac{N_1}{2}-1),...,2N-1,l=0,1,...,\frac{M_1}{2},k'=k-N \text{ i } l'=l \\ k=0,1,...,\frac{N_1}{2},l=2M-(\frac{M_1}{2}-1),...,2M-1,k'=k \text{ i } l'=l-M \\ k=2N-(\frac{N_1}{2}-1),...,2N-1,l=2M-(\frac{M_1}{2}-1),...,2M-1,k'=k-N \text{ i } l'=l-M \end{array}\right\},$$

ϵ to dopuszczalna wartość błędu w danej metryce, natomiast moduł z różnicy wartości kubatur $X_{2N\cdot 2M}^{2D}(k,l)$ oraz $X_{N\cdot M}^{2D}(k',l')$ rozumiany jest jako moduł liczy zespolonej. Bez utraty ogólności rozważań w powyższym wzorze można przyjąć za $T_1 = 1$ i $T_2 = 1$, tzn.

$$\epsilon_{MD}^{2DO}(k,l) = \max_{B}\left\{\frac{1}{3}\|X_{2N\cdot 2M}^{2D}(k,l) - X_{N\cdot M}^{2D}(k',l')\|\right\} \leqslant \epsilon. \tag{3.38}$$

Z wykorzystaniem wyrażenia (3.38) możliwa jest wówczas budowa algorytmu adaptacyjnego obliczania dwuwymiarowego przekształcenia Fouriera (ADFT2D).

ALGORYTM 3.3 *(Algorytm adaptacyjnego obliczania dwuwymiarowego przekształcenia Fouriera)*
Na wejściu dany musi być sygnał $x(t,s)$ określony na przedziale $[0,1]\times[0,1]$ i przyjmujący wartości rzeczywiste, zbiór jego N na M próbek pobranych w punktach (t_n, s_m), gdzie $t_n = n\Delta t$ i $\Delta t = 1/N$, oraz $s_m = m\Delta s$ i $\Delta s = 1/M$, dopuszczalna wartość błędu ϵ, a także liczba współczynników widmowych $N_1 \leqslant N$ na $M_1 \leqslant M$ (N_1 i M_1 parzyste), które brane są pod uwagę w procesie adaptacji.

1. *Oblicz $X_{N\cdot M}^{2D}(k,l)$ dla $k = 0,1,...,\frac{N_1}{2}$, $l = 0,1,...,\frac{M_1}{2}$.*
2. *Dobierz N na M próbek w punktach (t_{2n+1}, s_{2m+1}) dla $n = 0,1,...,N-1$ oraz $m = 0,1,...,M-1$, gdzie $t_{2n+1} = (2n+1)\Delta t/2$ i $s_{2m+1} = (2m+1)\Delta s/2$.*

3. Na podstawie zbioru $2N$ na $2M$ próbek oblicz:

$$X^{2D}_{2N\cdot 2M}(k,l) \text{ dla } k = 0,1,...,\tfrac{N_1}{2} \text{ i } l = 0,1,...,\tfrac{M_1}{2},$$

$$X^{2D}_{2N\cdot 2M}(2N-k,l) \text{ dla } k = 1,2,...,\tfrac{N_1}{2}-1 \text{ i } l = 0,1,...,\tfrac{M_1}{2},$$

$$X^{2D}_{2N\cdot 2M}(k,2M-l) \text{ dla } k = 0,1,...,\tfrac{N_1}{2} \text{ i } l = 1,2,...,\tfrac{M_1}{2}-1, \text{ oraz}$$

$$X^{2D}_{2N\cdot 2M}(2N-k,2M-l) \text{ dla } k = 1,2,...,\tfrac{N_1}{2}-1 \text{ i } l = 1,2,...,\tfrac{M_1}{2}-1.$$

4. Oblicz przybliżoną wartość ϵ^{2DO}_{MD} na podstawie następującej zależności

$$\epsilon^{2DO}_{MD}(k,l) = \max_B \left\{ \frac{1}{3} \| X^{2D}_{2N\cdot 2M}(k,l) - X^{2D}_{N\cdot M}(k',l') \| \right\}.$$

Jeżeli $\epsilon^{2DO}_{MD} \leqslant \epsilon$ to skocz do 6.

5. Podstaw $X^{2D}_{N\cdot M}(k,l) = X^{2D}_{2N\cdot 2M}(k,l)$ dla wszystkich par (k,l), dla których obliczono $X^{2D}_{2N\cdot 2M}(k,l)$, $N = N\cdot 2$, $M = M\cdot 2$ oraz $\Delta t = \Delta t/2$ i $\Delta s = \Delta s/2$. Skocz do 2.

6. Wypisz:

$$X^{2D}_{2N\cdot 2M}(k,l) \text{ dla } k = 0,1,...,\tfrac{N_1}{2} \text{ i } l = 0,1,...,\tfrac{M_1}{2},$$

$$X^{2D}_{2N\cdot 2M}(2N-k,l) \text{ dla } k = 1,2,...,\tfrac{N_1}{2}-1 \text{ i } l = 0,1,...,\tfrac{M_1}{2},$$

$$X^{2D}_{2N\cdot 2M}(k,2M-l) \text{ dla } k = 0,1,...,\tfrac{N_1}{2} \text{ i } l = 1,2,...,\tfrac{M_1}{2}-1,$$

$$X^{2D}_{2N\cdot 2M}(2N-k,2M-l) \text{ dla } k = 1,2,...,\tfrac{N_1}{2}-1 \text{ i } l = 1,2,...,\tfrac{M_1}{2}-1,$$

oraz wartości $2N$, $2M$ i ϵ^{2DO}_{MD}.

Koniec.

Powyższy algorytm umożliwia adaptacyjny dobór liczby $2N$ na $2M$ próbek sygnału, która wystarcza do obliczenia N_1 na M_2 niskoczęstotliwościowych współczynników widmowych z błędem mniejszym lub równym dopuszczalnej w metryce Czebyszewa wartości ϵ.

Dla dyskretnych dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju, które to przekształcenia stanowią kubatury numerycznego obliczania całkowego kosinusowego i sinusowego przekształcenia Fouriera, wyrażenia służące do automatycznej kontroli doboru liczby próbek, przyjmą następujące postaci

$$\epsilon^{P2DOII}_{2N\cdot 2M}(k,l) = \max_{\substack{k=0,1,...,N-1 \\ l=0,1,...,M-1}} \left\{ \frac{1}{3p_k p_l} \left| X^{P2DII}_{2N\cdot 2M}(k,l) - X^{P2DII}_{N\cdot M}(k,l) \right| \right\} \leqslant \epsilon \qquad (3.39)$$

dla przekształceń drugiego rodzaju, oraz

$$\epsilon_{2N\cdot 2M}^{P2DOIV}(k,l) = \max_{\substack{k=0,1,...,N-1\\ l=0,1,...,M-1}} \left\{ \frac{1}{3} \left| X_{2N\cdot 2M}^{P2DIV}(k,l) - X_{N\cdot M}^{P2DIV}(k,l) \right| \right\} \leqslant \epsilon \tag{3.40}$$

dla przekształceń rodzaju czwartego, gdzie ϵ to dopuszczalna wartość błędu w metryce Czebyszewa, natomiast P to typ przekształcenia, tj. przekształcenie kosinusowe (C) lub sinusowe (S). Ponadto bez utraty ogólności w powyższych wzorach przyjęto za $T_1 = 1$ oraz $T_2 = 1$. Tym samym możliwe jest skonstruowanie algorytmu adaptacyjnego dla wskazanych dwuwymiarowych przekształceń trygonometrycznych.

ALGORYTM 3.4 *(Algorytm adaptacyjnego obliczania dwuwymiarowych kosinusowych i sinusowych przekształceń Fouriera)*
Na wejściu dany musi być sygnał $x(t,s)$ określony na przedziale $[0,1] \times [0,1]$, zbiór jego N na M próbek pobranych w punktach (t'_n, s'_m), gdzie $t'_n = (n+1/2)\Delta t$ oraz $s'_m = (m+1/2)\Delta s$ dla $\Delta t = 1/N$ i $\Delta s = 1/M$, liczba współczynników widmowych $N_1 \leqslant N$ na $M_1 \leqslant M$, które brane są pod uwagę w procesie adaptacji, typ przekształcenia P, oraz dopuszczalna wartość błędu ϵ.

1. *Oblicz $X_{N\cdot M}^{P2DII}(k,l)$ $(X_{N\cdot M}^{P2DIV}(k,l))$ dla $k = 0,1,...,N_1-1$ oraz $l = 0,1,...,M_1-1$.*

2. *Pobierz $2N$ na $2M$ próbek w punktach (t'_n, s'_m), określonych przez dyskretne wartości $t'_n = (n+1/2)\Delta t/2$ i $s'_m = (m+1/2)\Delta s/2$ dla $n = 0,1,...,2N-1$ oraz $m = 0,1,...,2M-1$.*

3. *Na podstawie zbioru $2N$ na $2M$ próbek oblicz $X_{2N\cdot 2M}^{P2DII}(k,l)$ $(X_{2N\cdot 2M}^{P2DIV}(k,l))$ dla $k = 0,1,...,N_1-1$ i $l = 0,1,...,M_1-1$.*

4. *Oblicz przybliżoną wartość ϵ_{MD}^{P2DOII} (ϵ_{MD}^{P2DOIV}) na podstawie następującej zależności*

$$\epsilon_{MD}^{P2DOII} = \max_{\substack{k=0,1,...,N_1-1\\ l=0,1,...,M_1-1}} \left\{ \frac{1}{3p_k p_l} \left| X_{2N\cdot 2M}^{P2DII}(k,l) - X_{N\cdot M}^{P2DII}(k,l) \right| \right\}$$

$$(\epsilon_{MD}^{P2DOIV} = \max_{\substack{k=0,1,...,N_1-1\\ l=0,1,...,M_1-1}} \left\{ \frac{1}{3} \left| X_{2N\cdot 2M}^{P2DIV}(k,l) - X_{N\cdot M}^{P2DIV}(k,l) \right| \right\}).$$

   Jeżeli ϵ_{MD}^{OPII} $(\epsilon_{MD}^{OPIV}) \leqslant \epsilon$ to skocz do 6.

5. *Dokonaj podstawienia $X_{N\cdot M}^{P2DII}(k,l) = X_{2N\cdot 2M}^{P2DII}(k,l)$ $(X_{N\cdot M}^{P2DIV}(k,l) = X_{2N\cdot 2M}^{P2DIV}(k,l))$ dla $k = 0,1,...,N_1-1$ i $l = 0,1,...,M_1-1$, $N = N\cdot 2$, $M = M\cdot 2$ oraz $\Delta t = \Delta t/2$ i $\Delta s = \Delta s/2$. Skocz do 2.*

6. *Wypisz $X_{2N\cdot 2M}^{P2DII}(k,l)$ $(X_{2N\cdot 2M}^{P2DII}(k,l))$ dla $k = 0,1,...,N_1-1$ i $l = 0,1,...,M_1-1$, oraz wartości $2N$, $2M$ i ϵ_{MD}^{P2DOII} (ϵ_{MD}^{P2DOIV}).*

Powyższy algorytm pozwala na dobór liczby $2N$ na $2M$ próbek sygnału, które wystarczają do obliczenia N_1 na M_1 niskoczęstotliwościowych współczynników widmowych wybranego (zależnego od P) dwuwymiarowego przekształcenia Fouriera, z błędem nie większym niż dopuszczalna wartość ϵ, wyrażona w metryce Czebyszewa. W zależności od rodzaju przekształcenia dyskretnego algorytmy adaptacyjne oznaczać będziemy jako ADCT2D-II (ADCT2D-IV) dla przekształceń kosinusowych, oraz ADST2D-II (ADST2D-IV) dla przekształceń sinusowych.

Otrzymane wyrażenia (patrz (3.38) - (3.40)) kontrolujące dobór liczby próbek sygnału mają charakter heurystyczny. Ich skuteczność została zweryfikowana w sposób empiryczny w rozdziale 7.

3.5. Podsumowanie i wnioski

Na wstępie niniejszego rozdziału przedstawiono sposób konstruowania kwadratur numerycznych pierwszego rzędu, które pozwalają na przybliżone obliczanie wartości całek, a także podano wzory umożliwiające wyznaczenie rzeczywistych wartości błędów obliczania całek przy pomocy rozpatrywanych kwadratur. Przedstawione podejście rozszerzono następnie na przypadki całek z funkcji dwóch zmiennych, oraz kubatur całkowania numerycznego.

W dalszej części rozdziału wykazano, iż dyskretne przekształcenia Fouriera w przypadku jednowymiarowym stanowią kwadratury (odpowiednio kubatury dla funkcji dwóch zmiennych) numerycznego obliczania przekształceń Fouriera w postaci całkowej. Otrzymane w ten sposób wyniki pozwoliły na przeniesienie na grunt przekształceń trygonometrycznych znanych metod adaptacyjnego całkowania numerycznego [7, 39], co z kolei dało możliwość skonstruowania algorytmów adaptacyjnych dla wspomnianych jedno- i dwuwymiarowych przekształceń trygonometrycznych. Istotą proponowanych algorytmów adaptacyjnych jest możliwość automatycznego doboru takiej liczby próbek sygnału, która jest wystarczająca do obliczenia przekształcenia całkowego z żądaną dokładnością, tutaj wyrażoną w metryce Czebyszewa. Propozycje algorytmów adaptacyjnych dla wszystkich rozważanych przekształceń zamieszczono w ostatnich dwóch sekcjach niniejszego rozdziału.

Stąd na podstawie przeprowadzonych rozważań oraz otrzymanych wyników można stwierdzić, iż

- Możliwa jest budowa algorytmów adaptacyjnych dla przekształceń trygonometrycznych Fouriera oraz kosinusowego i sinusowego przekształcenia Fouriera, które dają możliwość obliczania reprezentacji sygnałów w bazach tych przekształceń, zgodnie z kryterium *dokładności reprezentacji - czasu realizacji obliczeń.*

- W świetle przyjętych założeń, zaproponowane wyrażenia kontrolujące dobór liczby próbek w kolejnych etapach adaptacji, mają charakter heurystyczny.

- Mając na uwadze charakter przyjętej heurystyki, a także sposób jej wyznaczenia, można wnioskować, iż analogiczne algorytmy adaptacyjne da się skonstruować dla przypadków więcej niż dwuwymiarowych przekształceń trygonometrycznych.

Zgodnie z zaproponowanymi algorytmami adaptacyjnymi, proces obliczania kwadratury (kubatury) w postaci $2N$ punktowego ($2N$ na $2M$ punktowego) przekształcenia

dyskretnego jest na każdym etapie adaptacji realizowany na podstawie całego dostępnego zbioru $2N$ próbek ($2N$ na $2M$ próbek). Zatem do obliczania wartości kwadratury (kubatury) na bieżącym etapie nie jest wykorzystywana wartość kwadratury (kubatury) obliczonej na etapie poprzedzającym. Takie podejście daje złożoność obliczeniową rzędu $\mathcal{O}(N^2)$ (rzędu $\mathcal{O}(N^4)$ dla przypadku dwuwymiarowego) dla każdego etapu adaptacji. W tym miejscu należy zadać pytanie: czy możliwa jest redukcja złożoności zaproponowanych algorytmów adaptacyjnych, np. poprzez zastosowanie szybkich algorytmów obliczania dyskretnych przekształceń trygonometrycznych? Odpowiedź na to pytanie jest twierdząca. Do tego celu wymagane są jednak szybkie algorytmy, które umożliwią obliczanie przekształcenia $2N$ punktowego ($2N$ na $2M$ punktowego) bezpośrednio na bazie przekształcenia N punktowego (N na M punktowego). Algorytmy takie noszą miano szybkich algorytmów z przerzedzeniem w czasie [52]. Wzory rozkładów takich algorytmów dla dyskretnych (jedno- i dwuwymiarowych) przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju zostały opisane w rozdziale 4 niniejszej książki. Algorytmy te dają możliwość budowy szybkich algorytmów adaptacyjnych, które zaprezentowane zostały w rozdziale 5.

Dodatkowo w rozdziale 6 zamieszczono wyrażenia oceny błędu w innych, niż metryka Czebyszewa, popularnych metrykach wykorzystywanych w zadaniach cyfrowego przetwarzania danych.

4

Szybkie algorytmy dla przekształceń dyskretnych

W rozdziale 3 przedstawione zostały propozycje algorytmów adaptacyjnego obliczania jedno- i dwuwymiarowych przekształceń Fouriera. Przy czym adaptacyjne obliczanie przekształceń trygonometrycznych realizowane jest w oparciu o kwadratury całkowania numerycznego. Dla rozważanych przekształceń całkowych jako kwadratury numeryczne wybrano dyskretne jedno- i dwuwymiarowe przekształcenia Fouriera oraz kosinusowe i sinusowe przekształcenia drugiego i czwartego rodzaju (patrz rozdział 3). Zaproponowane podejście adaptacyjne umożliwia automatyczny dobór liczby próbek sygnału wejściowego, która jest zależna od żądanej dokładności numerycznego obliczania widma. Złożoność obliczeniowa zaproponowanych algorytmów adaptacyjnych jest rzędu $\mathcal{O}(N^2)$ dla przekształceń jednowymiarowych, oraz rzędu $\mathcal{O}(N^4)$ dla przypadku dwuwymiarowego.

Jak już wspomniano w rozdziale 3, istnieje możliwość redukcji złożoności obliczeniowej algorytmów adaptacyjnych poprzez zastosowanie szybkich algorytmów obliczania dyskretnych przekształceń trygonometrycznych. Dla przypadku jednowymiarowego wymagane szybkie algorytmy powinny umożliwić obliczanie kwadratur wyznaczanych w danych kroku adaptacji z wykorzystaniem ich wartości z etapu poprzedzającego. Wymóg ten sprowadza się do obliczania przekształcenia $2N$-punktowego, które odpowiada kwadraturom wyznaczanym na danym etapie adaptacji, bezpośrednio na podstawie przekształcenia N-punktowego, które obliczono na etapie poprzedzającym. Własność taką posiadają tzw. szybkie algorytmy z przerzedzeniem w czasie. W przypadku dwuwymiarowym stosuje się analogiczne podejście.

Gwałtowny rozwój metod syntezy szybkich algorytmów obliczania dyskretnego przekształcenia Fouriera zapoczątkowała praca [28] z 1965 r. W pracy tej zaprezentowano szybki algorytm o złożoności $\mathcal{O}(Nlog_2N)$, podczas gdy w latach wcześniejszych (poza nielicznymi autorami (patrz rozdział 1), których prace nie zyskały należnej popularności), bariera $\mathcal{O}(N^2)$ wydawała się nie do przezwyciężenia. Algorytm przedstawiony w artykule [28] jest do dnia dzisiejszego najpopularniejszym szybkim algorytmem obliczania przekształcenia Fouriera. Zakłada on obliczanie przekształcenia N-punktowego dla $N = N_1N_2$ na podstawie N_1 przekształceń o liczbie N_2 punktów. Jeżeli parametr N_2 można przedstawić w postaci iloczynu dwóch liczb, to dany schemat można stosować

rekurencyjnie. Uzyskiwana złożoność obliczeniowa jest rzędu $\mathcal{O}(Nlog_2N)$. W przypadku N będącego całkowitą potęgą dwóch, oraz $N_1 = 2$, otrzymujemy tak zwany szybki algorytm typu radix-2.

Kolejne lata to prace nad poszukiwaniem bardziej efektywnych algorytmów, przy czym rząd obliczeń $\mathcal{O}(Nlog_2N)$ do dziś dnia pozostaje ten sam. Jako przykłady można podać: szybki algorytm typu radix-4 [11, 121] dla $N = 2^p$ i parzystych p - uzyskano przyspieszenie o około 25% w stosunku do radix-2, schemat obliczania dyskretnego przekształcenia Fouriera za pomocą transformaty świergotowej - możliwość sprzętowej realizacji obliczeń przy użyciu filtrów splotowych [13, 105], algorytm Rader'a-Brenner'a [106] o zredukowanej liczbie mnożeń kosztem większej liczby dodawań, to także algorytm Bruun'a [18] o arytmetyce rzeczywistej dla sygnałów przyjmujących wartości rzeczywiste, algorytm Winograda [149] dla $N = N_1N_2...N_d$, gdzie N_i dla $i = 1, 2, ..., d$ są parami pierwsze - znacząca redukcja dodawań i mnożeń (złożoność liniowa), czy też algorytm "split-radix 2/4" [35] łączący cechy algorytmów radix-2 i radix-4, tzn. $N = 2^p$ dla p całkowitych oraz przyspieszenie obliczeń o około 25% w porównaniu z radix-2. To również techniki obliczania wybranych prążków widma Fouriera (rekurencyjny algorytm Goertzela), rekurencyjnego wyznaczania dyskretnego przekształcenia Fouriera w chwili, gdy zmianie ulega jedynie pierwsza (lub ostatnia) próbka sygnału [161], oraz metoda obliczania przekształcenia $2N$-punktowego dla sygnału rzeczywistego, przy użyciu przekształcenia N-punktowego (w jęz. ang. *Packing Algorithm*) [26, 122].

Ostatnie klika lat to publikacje zawierające między innymi propozycje: metod automatycznej generacji szybkich algorytmów dla przekształceń trygonometrycznych [103], algorytmu o liniowej złożoności multiplikatywnej i prostszej strukturze niż algorytm Winograda [3], czy też efektywne implementacje programowe znanych już algorytmów dla nowoczesnych procesorów, np. z wykorzystaniem instrukcji jednoczesnego mnożenia i dodawania FMA (ang. *Fused Multiply-Add Instruction*) [87].

Wiele ze znanych szybkich algorytmów obliczania dyskretnego przekształcenia Fouriera spełnia wspomniane wymagania algorytmów adaptacyjnych, są to między innymi algorytmy typu radix-2 [28] i "split-radix 2/4" [35], które wybrano w niniejszej monografii, a ich wzory rozkładu zamieszczono w sekcji 4.1.1.

Zagadnieniom syntezy szybkich algorytmów dla dyskretnych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju poświęcono równie wiele uwagi. Algorytmy te można najogólniej podzielić na dwie grupy: algorytmy pośrednie, które wykorzystują szybkie algorytmy innych przekształceń, np.: Fouriera [5, 46, 47, 86, 152], Walsha-Hadamarda [151], czy też przekształcenia Hartleya [76], oraz algorytmy bezpośrednie, oparte o faktoryzację macierzy przekształceń kosinusowych i sinusowych, np. [22, 66, 134, 145, 157], które cechuje mniejsza złożoność obliczeniowa niż algorytmy z pierwszej grupy. Ostatnie lata to między innymi prace nad: algorytmami dwuetapowymi [53, 150, 156] o prostych strukturach, dających redukcję wymagań dla implementacji sprzętowych, algorytmy rekurencyjne obliczające przekształcenia N-punktowe na bazie wielu przekształceń o niewielkiej (różnej) liczbie punktów (ang. *Paired Transforms*) [44] - uzyskano znaczne redukcje operacji arytmetycznych, to również algorytmy przybliżone oparte o arytmetykę stałopozycyjną, o mniejszych wymaganiach w stosunku do ich implementacji sprzętowych [49, 104, 111], czy też algorytmy dedykowane dla komputerów kwantowych [143].

Dla dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju, wymagane z punktu widzenia rozważanym algorytmów adaptacyjnych szybkie

struktury obliczeniowe z przerzedzeniem w czasie, zaproponowane zostały w pracach [54, 55] odpowiednio dla przypadku sygnałów jedno- i dwuwymiarowych. Za punkt wyjścia przyjęto zaprezentowane w artykule [156] szybkie algorytmy dwuetapowe dla przekształceń: DCT-II, DCT-IV, DST-II oraz DST-IV, a następnie na ich podstawie skonstruowano algorytmy z przerzedzeniem w czasie, jedynie poprzez zastosowanie innej niż *bit-reverse* [73] permutacji elementów wejściowego ciągu danych. Stąd proponowane szybkie algorytmy z przerzedzeniem w czasie charakteryzuje taka sama złożoność obliczeniowa co znane algorytmy dwuetapowe, tzn. złożoność obliczeniowa rzędu $\mathcal{O}(Nlog_2N)$.

W sekcji 4.1.2 zamieszczono wzory rozkładów szybkich algorytmów dwuetapowych, natomiast w sekcji 4.1.3 znaleźć można wyprowadzenia szybkich algorytmów z przerzedzeniem w czasie dla przekształceń: DCT-II, DCT-IV, DST-II oraz DST-IV. Przez wzgląd na bezpośrednie analogie pomiędzy rozważanymi przypadkami, wyprowadzenia poszukiwanych szybkich algorytmów zademonstrowano wyłącznie na przykładzie przekształcenia kosinusowego drugiego rodzaju. Ponadto dla wszystkich przedstawionych szybkich algorytmów zamieszczono wyrażenia pozwalające na dokładne określenie liczby operacji arytmetycznych w postaci dodawań (odejmowań) i mnożeń, których liczba jest zależna od długości przekształcenia.

W praktyce do szybkiego obliczania przekształceń dwuwymiarowych często stosuje się tzw. podejście wierszowo-kolumnowe oparte o własność separowalności całkowych przekształceń Fouriera. Podejście to wykorzystuje szybkie algorytmy dla przekształceń jednowymiarowych, które stosuje się początkowo do każdego z wierszy (każdej z kolumn), a następnie do każdej z kolumn (każdego z wierszy) macierzy próbek sygnału wejściowego [154]. Metoda ta nie może być jednak wykorzystana do szybkiego obliczania przekształceń całkowych w oparciu o proponowane schematy adaptacyjne (patrz rozdział 3). W tym przypadku, podobnie jak dla przekształceń jednowymiarowych, wymagane są szybkie algorytmy z przerzedzeniem w czasie, które umożliwią obliczanie przekształcenia $2N$ na $2M$-punktowego, odpowiadającego kwadraturze wyznaczanej na danym etapie adaptacji, na bazie przekształcenia N na M-punktowego, będącego kwadraturą pozyskaną w poprzednim kroku. W przypadku przekształcenia Fouriera wymagany szybki algorytm został opisany w pracy [8]. Wzór rozkładu tego algorytmu zamieszczono w sekcji 4.2.1. Dla dyskretnych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju algorytmy takie opisano w pracy [55]. Wzory rozkładu tych algorytmów, wraz z wzorami na dokładne liczby wymaganych operacji dodawań (odejmowań) i mnożeń, zostały zamieszczone w sekcji 4.2.3 monografii.

W dodatku A zestawiono procedury napisane w języku C, które zawierają implementacje wszystkich przedstawionych szybkich algorytmów, włącznie z procedurami realizującymi wymagane permutacje elementów wejściowych ciągów danych.

4.1. Szybkie algorytmy dla dyskretnych przekształceń jednowymiarowych

4.1.1. Szybkie algorytmy dla dyskretnego przekształcenia Fouriera

W niniejszej sekcji przedstawione zostaną wzory rozkładów szybkich algorytmów z przerzedzeniem w czasie dla dyskretnego jednowymiarowego przekształcenia Fouriera, typu radix-2 [28] oraz "split-radix 2/4" [35].

Algorytm z przerzedzeniem w czasie typu radix-2

W pracy [28] przedstawiony został jeden z najpopularniejszych algorytmów szybkiego obliczania dyskretnego przekształcenia Fouriera (FFT z ang. *Fast Fourier Transform*). Nie należy on do najszybszych znanych algorytmów, jednakże charakteryzuje go prostota i łatwość implementacji w dowolnym języku programowania. Algorytm ten należy do klasy algorytmów z przerzedzeniem w czasie typu radix-2. Dokładny jego opis wraz z wyprowadzeniem znaleźć można na przykład w pracach [28, 73, 88].

Zacznijmy od przypomnienia definicji dyskretnego przekształcenia Fouriera przedstawionej w rozdziale 2. Niech $x(n)$ oznacza N elementowy ciąg liczb rzeczywistych, który reprezentuje próbki sygnału wejściowego $x(t)$ pobierane w dyskretnych chwilach czasowych t_n, gdzie N jest liczbą naturalną, będącą potęgą dwóch. Założenie takie jest niezbędne w przypadku szybkich algorytmów typu radix-2 i wynika z rekurencyjnego podziału ciągu wejściowego na elementy o indeksach parzystych i nieparzystych. Wówczas dyskretne przekształcenie Fouriera definiujemy jako

$$X_N(k) = DFT_N\{x(n)\} \triangleq \frac{1}{N}\sum_{n=0}^{N-1} x(n)W_N^{kn}, \tag{4.1}$$

gdzie

$$W_N^{kn} = e^{-i\frac{2\pi}{N}kn} = cos(\frac{2\pi}{N}kn) - isin(\frac{2\pi}{N}kn)$$

dla $k = 0, 1, ..., N-1$. Zgodnie z pracą [28] wzór rozkładu szybkiego algorytmu typu radix-2 z przerzedzeniem w czasie dla jednowymiarowego przekształcenia Fouriera (4.1) przyjmie postać wyrażenia

$$\begin{aligned} X_N(k) &= X_0(k) + X_1(k)W_N^k, \\ X_N(k+\tfrac{N}{2}) &= X_0(k) - X_1(k)W_N^k \end{aligned} \tag{4.2}$$

dla $k = 0, 1, ..., N/2-1$, gdzie

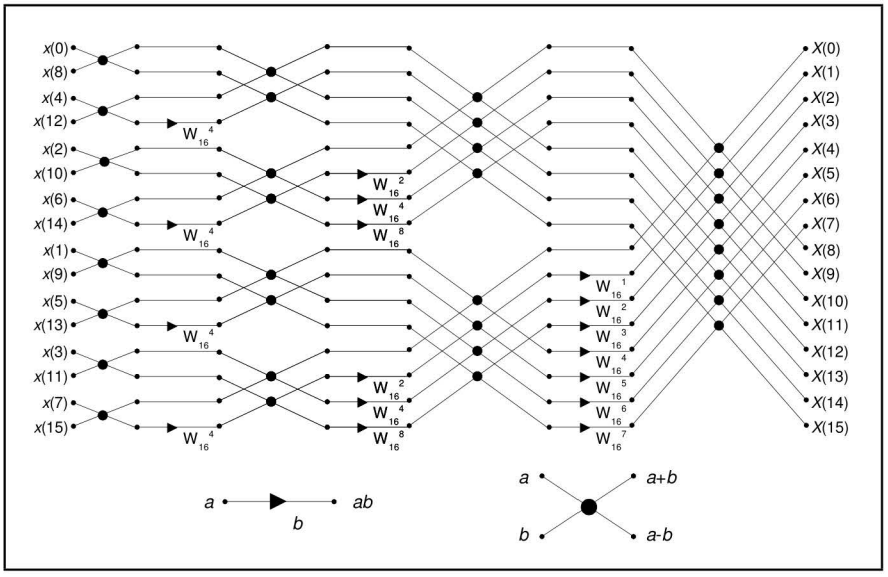
$$X_0(k) = DFT_{\frac{N}{2}}\{x(2n)\},$$

$$X_1(k) = DFT_{\frac{N}{2}}\{x(2n+1)\}.$$

Z równania (4.2) widać, iż według algorytmu radix-2 przekształcenie N-punktowe obliczane jest na bazie dwóch przekształceń $N/2$-punktowych, które operują odpowiednio na parzystych i nieparzystych elementach wejściowego ciągu danych, przy czym wyniki otrzymane po przekształceniu nieparzystych elementów wymnażane są dodatkowo poprzez czynnik W_N^k.

Jeżeli przyjmiemy, że przekształcenie $X_N(k)$ odpowiada kwadraturze N-punktowej, to przekształcenie $X_0(k)$ będzie kwadraturą $N/2$-punktową obliczoną na poprzednim etapie adaptacji. Stąd szybki algorytm obliczania DFT opisany zależnością (4.2), może posłużyć do budowy szybkiego algorytmu adaptacyjnego obliczania całkowego przekształcenia Fouriera.

Rekurencyjne użycie wzoru (4.2) umożliwia szybkie obliczenie przekształcenia DFT dla liczby punktów N będącej potęgą dwóch, tzn. spełniającej warunek $N = 2^s$, gdzie



Rysunek 4.1: Graf przepływowy szybkiego algorytmy typu radix-2 dla $N = 16$ punktowego DFT

s jest liczbą naturalną. Graf przepływowy szybkiego algorytmu typu radix-2 dla dyskretnego przekształcenia Fouriera w przypadku, gdy $N = 16$ przedstawiony został na rysunku 4.1. Dla przejrzystości zapisu kolejne współczynniki transformaty oznaczono przez $X(k)$ dla $k = 0, 1, ..., N-1$.

Ponieważ w każdym kroku rekurencyjnego obliczania przekształcenia zgodnie z wyrażeniem (4.2), elementy ciągu wejściowego grupowane są na elementy o indeksach parzystych i nieparzystych, to postępowanie takie wymusza pewną specyficzną permutację elementów transformowanego ciągu wejściowego $x(n)$ dla $n = 0, 1, ..., N-1$. Kolejność taka nosi nazwę $bit - reverse$ [73, 88] przez wzgląd na możliwość wyznaczenia nowego indeksu n dla elementu ciągu $x(n)$, poprzez odwrócenie kolejności zapisu bitów w rozwinięciu dwójkowym indeksu bieżącego. W tabeli 4.1 przedstawiono przyporządkowanie indeksom ich odpowiedników w kolejności $bit - reverse$, które jest charakterystyczne dla permutacji szesnastoelementowego ciągu danych.

Złożoność obliczeniowa algorytmu radix-2 jest rzędu $\mathcal{O}(Nlog_2N)$. Na wyznaczenie dokładnej liczby operacji rzeczywistych mnożeń Mn_N^{r2} i dodawań Do_N^{r2}, w zależności od długości przekształcenia, pozwalają wyrażenia (4.3) [88]

$$\begin{aligned} Mn_N^{r2} &= 2N\left(\log_2 N - 2\right) + 4, \\ Do_N^{r2} &= 3N\left(\log_2 N - 1\right) + N + 2. \end{aligned} \tag{4.3}$$

W dodatku A zamieszczono procedury napisane w języku C, które realizują zarówno przemieszanie zgodne z kolejnością $bit - reverse$ (procedura A.1), jak i szybkie dyskretne przekształcenie Fouriera według wzoru rozkładu (4.2) (procedura A.2).

Tabela 4.1: Przyporządkowanie indeksów w kolejności *bit-reverse* dla szesnastoelementowego ciągu danych

indeks	bit-reverse	indeks	bit-reverse
0000 (0)	0000 (0)	1000 (8)	0001 (1)
0001 (1)	1000 (8)	1001 (9)	1001 (9)
0010 (2)	0100 (4)	1010 (10)	0101 (5)
0011 (3)	1100 (12)	1011 (11)	1101 (13)
0100 (4)	0010 (2)	1100 (12)	0011 (3)
0101 (5)	1010 (10)	1101 (13)	1011 (11)
0110 (6)	0110 (6)	1110 (14)	0111 (7)
0111 (7)	1110 (14)	1111 (15)	1111 (15)

Algorytm z przerzedzeniem w czasie typu "split-radix 2/4"

Szybki algorytm typu split-radix zaproponowany w pracy [35] łączy w sobie cechy algorytmów radix-2 i radix-4. Dzięki czemu cechuje go złożoność obliczeniowa typowa dla algorytmu radix-4, tj. o 25% mniejsza niż radix-2, oraz możliwość obliczania przekształceń, których długości są potęgami dwóch, a nie czterech tak jak ma to miejsce w przypadku algorytmów typu radix-4. Tutaj przekształcenie N-punktowe obliczane jest na bazie jednego przekształcenia $N/2$-punktowego oraz dwóch przekształceń o długości $N/4$-punktów. Zgodnie z [35] wzór rozkładu szybkiego algorytmu "split-radix 2/4" z przerzedzeniem w czasie dla jednowymiarowego przekształcenia Fouriera opisuje wyrażenie

$$\begin{aligned} X_N(k) &= X_0(k) + X_2(k)W_N^k + X_3(k)W_N^{3k}, \\ X_N(k+\tfrac{N}{2}) &= X_0(k) - X_2(k)W_N^k - X_3(k)W_N^{3k} \end{aligned} \tag{4.4}$$

dla $k = 0, 1, ..., N/2 - 1$, gdzie

$$X_N(k) = DFT\{x(n)\},\ X_0(k) = DFT_{\frac{N}{2}}\{x(2n)\},$$

$$X_2(k) = DFT_{\frac{N}{4}}\{x(4n+1)\},\ X_3(k) = DFT_{\frac{N}{4}}\{x(4n+3)\}.$$

Z powyższej zależności wynika, że algorytm typu "split-radix 2/4" spełnia wymagania potrzebne do budowy szybkiego algorytmu adaptacyjnego obliczania przekształcenia Fouriera. Także i w tym przypadku, $X_N(k)$ odpowiada kwadraturze N-punktowej obliczanej na danym etapie adaptacji, natomiast $X_0(k)$ reprezentuje kwadraturę $N/2$-punktową obliczoną na etapie wcześniejszym. Dodatkowe przekształcenia wymagane do wyznaczenia $X_N(k)$, to dwa przekształcenia $N/4$-punktowe określone na elementach $x(4n+1)$ oraz $x(4n+3)$, których wartości wymnażane są odpowiednio poprzez

czynniki W_N^k i W_N^{3k}. Rekurencyjne użycie zależności (4.4) pozwala na szybkie obliczanie jednowymiarowego przekształcenia Fouriera o długości będącej potęgą dwóch, przy czym elementy wejściowego ciągu danych należy posortować zgodnie z kolejnością przemieszania $bit - reverse$.

Dokładną liczbę operacji rzeczywistych dodawań Do_N^{sr} i mnożeń Mn_N^{sr} dla algorytmu "split-radix 2/4" definiują poniższe zależności [35]

$$Mn_N^{sr} = N(\log_2 N - 3) + 4,$$

$$Do_N^{sr} = 3N(\log_2 N - 1) + 4.$$

W dodatku A zamieszczono procedurę napisaną w języku C, która realizuje szybki algorytm obliczania przekształcenia Fouriera wg. wzoru rozkładu (4.4) (patrz procedura A.2).

4.1.2. Szybkie algorytmy dwuetapowe dla dyskretnych przekształceń kosinusowych i sinusowych

Na treść niniejszej sekcji składają się zamieszczone w pracy [156] wzory rozkładów szybkich algorytmów dwuetapowych dla jednowymiarowych dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju.

Przekształcenie kosinusowe drugiego rodzaju

Jako pierwszy rozpatrzony zostanie przypadek szybkiego algorytmu dwuetapowego dla jednowymiarowego dyskretnego przekształcenia kosinusowego drugiego rodzaju (FCT-II). Ponieważ algorytm ten stanowi punkt wyjścia dla wyprowadzenia szybkiego algorytmu z przerzedzeniem w czasie, to w dalszej części paragrafu podane zostanie dokładne wyprowadzenie wzoru rozkładu dla tego algorytmu. Na początku przypomniana zostanie jednak definicja dyskretnego przekształcenia kosinusowego drugiego rodzaju.

Niech $x(n)$ oznacza N-elementowy ciąg liczb rzeczywistych będących próbkami transformowanego sygnału $x(t)$, które pobierane są w dyskretnych chwilach czasowych t'_n. Wówczas jednowymiarowe dyskretne przekształcenie kosinusowe drugiego rodzaju, określone dla ciągu $x(n)$, definiujemy jako

$$X_N^{CII}(k) = DCTII_N\{x(n)\} \triangleq \frac{p_k}{N}\left[\sum_{n=0}^{N-1} x(n)cos(\frac{\pi}{N}k(n+\frac{1}{2}))\right] \tag{4.5}$$

dla $k = 0, 1, ..., N-1$, gdzie

$$p_k = \begin{cases} 2 \text{ dla } k = 0, \\ \\ 1 \text{ dla } k \neq 0. \end{cases}$$

Ponieważ stałe $\frac{1}{N}$ i p_k nie wpływają na sposób wyprowadzenia szybkiego algorytmu i mnożenia przez nie można wykonać na samym końcu, to dla przejrzystości zapisu

zostaną one pominięte w dalszych przekształceniach. Wówczas rozpisując sumę w wyrażeniu (4.5) na dwie sumy odpowiednio po elementach o indeksach n parzystych i nieparzystych, otrzymujemy

$$\begin{aligned} X_N^{CII}(k) &= \sum_{n=0}^{N/2-1} x(2n)cos\left(\frac{\pi}{N}k(2n+\frac{1}{2})\right) + \\ &+ \sum_{n=0}^{N/2-1} x(2n+1)cos\left(\frac{\pi}{N}k(2n+\frac{3}{2})\right) = \\ &= \sum_{n=0}^{N/2-1} x(2n)cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2}-\frac{1}{4})\right) \\ &+ \sum_{n=0}^{N/2-1} x(2n+1)cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2}+\frac{1}{4})\right) = \\ &= \sum_{n=0}^{N/2-1} x(2n)cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2})-\frac{\pi}{2N}k\right) + \\ &+ \sum_{n=0}^{N/2-1} x(2n+1)cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2})+\frac{\pi}{2N}k\right). \end{aligned} \tag{4.6}$$

Następnie korzystając z własności kosinusa dla sumy kątów: $cos(\alpha+\beta) = cos(\alpha)cos(\beta)+sin(\alpha)sin(\beta)$, oraz dla różnicy kątów: $cos(\alpha-\beta) = cos(\alpha)cos(\beta)-sin(\alpha)sin(\beta)$, to wyrażenie (4.6) można zapisać w postaci

$$\begin{aligned} X_N^{CII}(k) &= \sum_{n=0}^{N/2-1} x(2n)\left[cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})\right)cos\left(\frac{\pi}{2N}k\right)+\right. \\ &+ \left. sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})\right)sin\left(\frac{\pi}{2N}k\right)\right] + \\ &+ \sum_{n=0}^{N/2-1} x(2n+1)\left[cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})\right)cos\left(\frac{\pi}{2N}k\right)-\right. \\ &- \left. sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})\right)sin\left(\frac{\pi}{2N}k\right)\right]. \end{aligned} \tag{4.7}$$

Stosując w tym miejscu Własność 2.4.1, zamieszczoną w rozdziale 2 i dotyczącą związków symetrii pomiędzy dyskretnym przekształceniem kosinusowym i sinusowym drugiego rodzaju, dalej zapisujemy wyrażenie (4.7) jako

$$\begin{aligned} X_N^{CII}(k) &= \sum_{n=0}^{N/2-1} x(2n)\left[cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})\right)cos\left(\frac{\pi}{2N}k\right)+\right. \\ &+ \left.(-1)^n cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(\frac{N}{2}-k)(n+\frac{1}{2})\right)sin\left(\frac{\pi}{2N}k\right)\right] + \end{aligned}$$

$$+ \sum_{n=0}^{N/2-1} x(2n+1)\left[cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2})\right)cos\left(\frac{\pi}{2N}k\right) - \right.$$

$$\left. - (-1)^n cos\left(\frac{\pi}{(\frac{N}{2})}(\frac{N}{2}-k)(n+\frac{1}{2})\right)sin\left(\frac{\pi}{2N}k\right)\right],$$

gdzie po odpowiednim zgrupowaniu składników otrzymujemy wyrażenie postaci

$$X_N^{CII}(k) = \left[\sum_{n=0}^{N/2-1}(x(2n)+x(2n+1))\,cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2})\right)\right]cos\left(\frac{\pi}{2N}k\right) + \tag{4.8}$$

$$+\left[\sum_{n=0}^{N/2-1}(-1)^n(x(2n)-x(2n+1))\,cos\left(\frac{\pi}{(\frac{N}{2})}(\frac{N}{2}-k)(n+\frac{1}{2})\right)\right]sin\left(\frac{\pi}{2N}k\right)$$

dla $k = 0, 1, ..., N/2-1$. W analogiczny sposób na podstawie własności (2.4.2) otrzymujemy

$$X_N^{CII}(N-k) = -\left[\sum_{n=0}^{N/2-1}(x(2n)+x(2n+1))\,cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2})\right)\right]sin\left(\frac{\pi}{2N}k\right) + \tag{4.9}$$

$$+\left[\sum_{n=0}^{N/2-1}(-1)^n(x(2n)-x(2n+1))\,cos\left(\frac{\pi}{(\frac{N}{2})}(\frac{N}{2}-k)(n+\frac{1}{2})\right)\right]cos\left(\frac{\pi}{2N}k\right)$$

dla $k = 0, 1, ..., N/2-1$. Wówczas wprowadzając dodatkowe oznaczenia postaci

$$X_0^{CII}(k) = \sum_{n=0}^{N/2-1}(x(2n)+x(2n+1))\,cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2})\right),$$

$$X_1^{CII}(k) = \sum_{n=0}^{N/2-1}[(-1)^n(x(2n)+x(2n+1))]\,cos\left(\frac{\pi}{(\frac{N}{2})}k(n+\frac{1}{2})\right),$$

oraz korzystając ze wzorów (4.8) i (4.9), możemy zapisać wzór rozkładu szybkiego dwuetapowego algorytmu obliczania dyskretnego przekształcenia kosinusowego drugiego rodzaju w ostatecznej postaci, za pomocą następujących zależności

$$X_N^{CII}(k) = X_0^{CII}(k)cos\left(\frac{\pi}{2N}k\right) + X_1^{CII}(\frac{N}{2}-k)sin\left(\frac{\pi}{2N}k\right)$$

dla $k = 1, 2, ..., N/2-1$,

$$X_N^{CII}(N-k) = X_1^{CII}(\frac{N}{2}-k)cos\left(\frac{\pi}{2N}k\right) - X_0^{CII}(k)sin\left(\frac{\pi}{2N}k\right) \tag{4.10}$$

dla $k = 1, 2, ..., N/2-1$ oraz

$$X_N^{CII}(0) = X_1^{CII}(0),\ X_N^{CII}(\frac{N}{2}) = \frac{\sqrt{2}}{2}X_2^{CII}(0).$$

Obliczanie przekształcenia N-punktowego zgodnie ze wzorem rozkładu szybkiego algorytmu dwuetapowego odbywa się z wykorzystaniem dwóch przekształceń $N/2$-punktowych, które nie operują bezpośrednio na elementach ciągu wejściowego, lecz na sumach i różnicach elementów o indeksach parzystych i nieparzystych. Stąd algorytm ten nie nadaje się do budowy szybkiego algorytmu adaptacyjnego dla przekształcenia kosinusowego.

Poprzez rekurencyjne użycie wyrażenia (4.10) otrzymujemy postać szybkiego algorytmu dwuetapowego dla przekształcenia o długości będącej całkowitą potęgą dwóch. Wówczas elementy wejściowego ciągu danych muszą być przemieszane zgodnie z kolejnością $bit - reverse$.

Z wzoru (4.10) wynika, iż szybki algorytm dwuetapowy DCT-II charakteryzuje złożoność obliczeniowa rzędu $\mathcal{O}(Nlog_2N)$. Dokładną liczbę operacji rzeczywistych mnożeń Mn_N^{CII} i dodawań Do_N^{CII} podają następujące wyrażenia

$$\begin{aligned} Mn_N^{CII} &= 2N\,(log_2N - 2) + N + 3, \\ Do_N^{CII} &= 2N\,(log_2N - 1) + 2. \end{aligned} \tag{4.11}$$

Wyprowadzenia szybkich algorytmów dwuetapowych dla dyskretnych przekształceń: sinusowego drugiego rodzaju oraz kosinusowego i sinusowego czwartego rodzaju przebiegają w sposób analogiczny do przypadku przekształcenia DCT-II. Zatem w dalszej części niniejszego rozdziału zostaną zamieszczone jedynie wzory rozkładów szybkich algorytmów dwuetapowych dla tych przekształceń.

Przekształcenie kosinusowe czwartego rodzaju

Dyskretne przekształcenie kosinusowe czwartego rodzaju dla N-elementowego ciągu liczb rzeczywistych $x(n)$ definiuje poniższa zależność (patrz rozdział 2)

$$X_N^{C(IV)}(k) = DCTIV_N\{x(n)\} \triangleq \frac{1}{N}\left[\sum_{n=0}^{N-1} x(n)cos(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2}))\right]$$

dla $k = 0, 1, ..., N-1$, gdzie $x(n)$ oznacza N-elementowy zbiór próbek sygnału $x(t)$, pobranych w chwilach t'_n. Pomijając czynnik normalizujący $\frac{1}{N}$, przez który mnożenie można wykonać na końcu, oraz stosując wyprowadzenie analogiczne do przypadku przekształcenia drugiego rodzaju, otrzymujemy dla przekształcenia DCT-IV wzór rozkładu szybkiego algorytmu dwuetapowego (FCT-IV) w postaci

$$\begin{aligned} X_N^{CIV}(k) &= X_0^{CIV}(k)cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) + \\ &+ X_1^{CIV}(\frac{N}{2} - k - \frac{1}{2})sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right), \end{aligned} \tag{4.12}$$

dla $k = 0, 1, ..., N/2-1$ oraz

$$\begin{aligned} X_N^{CIV}(N-k-1) &= X_1^{CIV}(\frac{N}{2} - k - \frac{1}{2})cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) - \\ &- X_0^{CIV}(k)sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 1$, gdzie $X_0^{CIV}(k)$ i $X_1^{CIV}(k)$ są określone jako

$$X_0^{CIV}(k) = \sum_{n=0}^{N/2-1} (x(2n) + x(2n+1)) \, cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right),$$

$$X_1^{CIV}(k) = \sum_{n=0}^{N/2-1} ((-1)^n (x(2n) - x(2n+1))) \, cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$, to $N/2$-punktowe przekształcenia kosinusowe czwartego rodzaju.

Na podstawie powyższej zależności widać, iż przekształcenie N-punktowe obliczane jest na bazie dwóch przekształceń $N/2$-punktowych operujących na sumach i różnicach elementów wejściowego ciągu danych. Zatem algorytm zbudowany zgodnie z zależnością (4.12) nie spełnia warunku wymaganego dla budowy szybkiego algorytmu adaptacyjnego, który zakłada, iż przekształcenie N-punktowe wyznacza się bezpośrednio na bazie przekształcenia $N/2$-punktowego, które jest odpowiednikiem kwadratury obliczanej w poprzednim kroku adaptacji, oraz na bazie zbioru $N/2$ próbek dobranych na danym etapie adaptacji.

Ze wzoru (4.12) wynika, iż złożoność obliczeniowa algorytmu dwuetapowego dla dyskretnego przekształcenia kosinusowego czwartego rodzaju jest rzędu $\mathcal{O}(Nlog_2N)$. Dokładną liczbę operacji rzeczywistych dodawań Do_N^{CIV} oraz mnożeń Mn_N^{CIV}, uzależnioną od długości przekształcenia N, można wyznaczyć przy pomocy następujących wyrażeń

$$\begin{aligned} Mn_N^{CIV} &= 2N\left(log_2N + \frac{1}{2}\right), \\ Do_N^{CIV} &= 2Nlog_2N. \end{aligned} \tag{4.13}$$

Poprzez rekurencyjne zastosowanie zależności (4.12) można obliczyć jedynie przekształcenie o długości N będącej całkowitą potęgą dwóch. Także i w tym przypadku elementy wejściowego ciągu danych muszą być przemieszane zgodnie z kolejnością $bit - reverse$.

Przekształcenie sinusowe drugiego rodzaju

Niech $x(n)$ oznacza N-elementowy ciąg liczb rzeczywistych będących próbkami sygnału wejściowego. Wówczas dyskretne przekształcenie sinusowe drugiego rodzaju definiujemy jako (patrz rozdział 2)

$$X_N^{SII}(k) = DSTII_N\{x(n)\} \triangleq \frac{p_k}{N}\left[\sum_{n=0}^{N-1} x(n) sin(\frac{\pi}{N}(k+1)(n+\frac{1}{2}))\right]$$

dla $k = 0, 1, ..., N-1$, gdzie

$$p_k = \begin{cases} 2 \text{ dla } k = 0, \\ \\ 1 \text{ dla } k \neq 0. \end{cases}$$

Wzór rozkładu szybkiego algorytmu (FST- II) dwuetapowego dla danego przekształcenia, z pominięciem stałej $\frac{1}{N}$ oraz p_k, przedstawia poniższa zależność

$$\begin{aligned} X_N^{SII}(k) &= X_0^{SII}(k)cos\left(\frac{\pi}{2N}(k+1)\right) \\ &- X_1^{SII}(\frac{N}{2}-k-2)sin\left(\frac{\pi}{2N}(k+1)\right), \end{aligned} \tag{4.14}$$

dla $k = 0, 1, ..., N/2-1$ oraz

$$\begin{aligned} X_N^{SII}(N-k-2) &= X_0^{SII}(\frac{N}{2}-k-2)sin\left(\frac{\pi}{2N}(k+1)\right) \\ &+ X_1^{SII}(k)cos\left(\frac{\pi}{2N}(k+1)\right) \end{aligned}$$

dla $k = 0, 1, ..., N/2-1$, gdzie

$$X_0^{SII}(k) = \sum_{n=0}^{N/2-1}(x(2n)+x(2n+1))\,sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right),$$

$$X_1^{SII}(k) = \sum_{n=0}^{N/2-1}((-1)^n(x(2n)+x(2n+1)))\,sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$.

Dwuetapowy algorytm FST-II charakteryzuje ta sama złożoność obliczeniowa co szybki algorytm dla przekształcenia kosinusowego drugiego rodzaju. Stąd, także i tym w przypadku, dokładną liczbę operacji rzeczywistych dodawań Do_N^{SII} i mnożeń Mn_N^{SII} podają zależności postaci

$$\begin{aligned} M_N^{SII} &= 2N\,(log_2N-2)+N+3, \\ D_N^{SII} &= 2N\,(log_2N-1)+2. \end{aligned} \tag{4.15}$$

Przekształcenie sinusowe czwartego rodzaju

Znów stosując analogiczne wyprowadzenie do pokazanego dla przypadku przekształcenia DCT-II, można wyznaczyć szybki algorytm dwuetapowy dla dyskretnego przekształcenia sinusowego czwartego typu (FST-IV). Przekształcenie to dla N-elementowego ciągu wejściowego $x(n)$ definiuje się jako (patrz rozdział 2)

$$X_N^{SIV}(k) = DSTIV_N\{x(n)\} \triangleq \frac{1}{N}\left[\sum_{n=0}^{N-1}x(n)sin(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2}))\right]$$

dla $k = 0, 1, ..., N-1$. Szybki algorytm dwuetapowy dla tego przekształcenia opisują natomiast zależności

$$\begin{aligned} X_N^{SIV}(k) &= X_0^{SIV}(k)cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) - \\ &- X_1^{SIV}(\frac{N}{2}-k-\frac{1}{2})sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right), \end{aligned} \tag{4.16}$$

dla $k = 0, 1, ..., N/2 - 1$,

$$\begin{aligned} X_N^{SIV}(N-k-1) &= X_0^{SIV}(k) sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) + \\ &+ X_1^{SIV}(\frac{N}{2}-k-\frac{1}{2}) cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 1$, gdzie

$$X_0^{SIV}(k) = \sum_{n=0}^{N/2-1} (x(2n) + x(2n+1)) \, sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right),$$

$$X_1^{SIV}(k) = \sum_{n=0}^{N/2-1} \left((-1)^n (x(2n) + x(2n+1))\right) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$, to $N/2$ - punktowe przekształcenia DST-IV określone na sumie i różnicy elementów wejściowego ciągu $x(n)$.

Ze wzoru (4.16) wynika, iż złożoność dwuetapowego algorytmu jest rzędu $\mathcal{O}(Nlog_2N)$. Dokładną liczbę operacji rzeczywistych dodawań Do_N^{SIV} i mnożeń Mn_N^{SIV} można wyznaczyć przy pomocy zależności

$$\begin{aligned} Mn_N^{CIV} &= 2N\left(log_2N + \frac{1}{2}\right), \\ Do_N^{CIV} &= 2Nlog_2N. \end{aligned} \tag{4.17}$$

Stosując wzór rozkładu w sposób rekurencyjny można zbudować szybki algorytm dwuetapowy dla dyskretnego przekształcenia sinusowego czwartego rodzaju o długości N będącej potęgą dwóch.

4.1.3. Szybkie algorytmy z przerzedzeniem w czasie dla dyskretnych przekształceń kosinusowych i sinusowych

W poprzedniej sekcji przedstawiono szybkie algorytmy dwuetapowe dla dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju. Jak wspomniano, algorytmy te nie posiadają struktur odpowiednich do budowy szybkich algorytmów adaptacyjnych. Stanowić jednak będą bazę dla wyprowadzeń szybkich algorytmów z przerzedzeniem w czasie, których wzory rozkładu przedstawione zostaną w niniejszej sekcji.

Ponieważ we wszystkich rozważanych przypadkach wyprowadzenia szybkich algorytmów z przerzedzeniem w czasie przebiegają w sposób zbliżony, to proces syntezy wspomnianych algorytmów zademonstrowany zostanie wyłącznie na przykładzie przekształcenia DCT-II. Dla pozostałych przypadków podane zostaną gotowe postaci wzorów rozkładu szybkich algorytmów bez ich wyprowadzeń.

Konstrukcja algorytmów dwuetapowych zakładała przemieszanie elementów ciągu $x(n)$, złożonego z N próbek sygnału wejściowego, zgodnie z kolejnością $bit - reverse$.

Tabela 4.2: Sposób przyporządkowania elementów $x(n)$ ciągom $a(n)$ oraz $b(n)$ dla przypadku $N = 16$

n	0	1	2	3	4	5	6	7
$a(n)$	$x(0)$	$x(3)$	$x(4)$	$x(7)$	$x(8)$	$x(11)$	$x(12)$	$x(15)$
$b(n)$	$x(1)$	$x(2)$	$x(5)$	$x(6)$	$x(9)$	$x(10)$	$x(13)$	$x(14)$

Wprowadźmy następujące $N/2$-elementowe ciągi pomocnicze złożone z pewnych kombinacji elementów $x(n)$, które definiujemy jako

$$\begin{aligned} a(n) &\triangleq \tfrac{1}{2}\left[(x(2n) + x(2n+1)) + (-1)^n (x(2n) - x(2n+1))\right], \\ b(n) &\triangleq \tfrac{1}{2}\left[(x(2n) + x(2n+1)) - (-1)^n (x(2n) - x(2n+1))\right] \end{aligned} \tag{4.18}$$

dla $n = 0, 1, ..., N/2 - 1$. Rozpisując powyższe wyrażenia otrzymamy równoważne zależności opisujące ciągi $a(n)$ i $b(n)$

$$a(n) = \begin{cases} x(2n) & \text{dla } n \text{ parzystych,} \\ x(2n+1) & \text{dla } n \text{ nieparzystych} \end{cases} \tag{4.19}$$

oraz

$$b(n) = \begin{cases} x(2n+1) & \text{dla } n \text{ parzystych,} \\ x(2n) & \text{dla } n \text{ nieparzystych,} \end{cases}$$

gdzie $n = 0, 1, ..., N/2-1$. Z zależności tych widać, iż elementami ciągów $a(n)$ oraz $b(n)$ są odpowiednio przemieszane elementy wejściowego ciągu $x(n)$. Ponadto łatwo wykazać, że ciągi te nie zawierają elementów wspólnych, tzn. elementów o tych samych indeksach, a zatem wykorzystują cały dostępny zbiór elementów $x(n)$ dla $n = 0, 1, ..., N-1$. W tabeli 4.2 pokazano przyporządkowanie elementów $x(n)$ nowo wprowadzonym ciągom pomocniczym $a(n)$ i $b(n)$ dla przypadku $N = 16$.

Należy tutaj zwrócić uwagę na dwie bardzo istotne z punktu widzenia dalszych rozważań własności ciągów (4.18). Własności te dotyczą sumy oraz różnicy elementów należących do wspomnianych ciągów i posiadających te same indeksy n. Mianowicie

$$\begin{aligned} a(n) + b(n) &= \frac{1}{2}\left[(x(2n) + x(2n+1)) + (-1)^n (x(2n) - x(2n+1))\right] + \\ &+ \frac{1}{2}\left[(x(2n) + x(2n+1)) - (-1)^n (x(2n) - x(2n+1))\right] = \\ &= x(2n) + x(2n+1) \end{aligned} \tag{4.20}$$

oraz

$$\begin{aligned} a(n) - b(n) &= \frac{1}{2}\left[(x(2n) + x(2n+1)) + (-1)^n (x(2n) - x(2n+1))\right] - \\ &- \frac{1}{2}\left[(x(2n) + x(2n+1)) - (-1)^n (x(2n) - x(2n+1))\right] = \\ &= (-1)^n (x(2n) - x(2n+1)) \end{aligned} \quad (4.21)$$

dla $n = 0, 1, ..., N/2-1$. Zatem sumy oraz różnice elementów $a(n)$ i $b(n)$ o tych samych indeksach odpowiadają kombinacjom elementów ciągu $x(n)$, na których określone były przekształcenia X_0 i X_1 we wzorach rozkładu szybkich algorytmów dwuetapowych. W przeciwieństwie do kombinacji elementów $x(n)$ przekształcenia określone na sumie i różnicy elementów $a(n)$ i $b(n)$, zgodnie z własnością liniowości rozważanych przekształceń, można rozpisać jako sumy i różnice przekształceń opartych bezpośrednio na elementach $a(n)$ i $b(n)$, tj. na elementach ciągu $x(n)$ (patrz wzór (4.19)). Ta właściwość jest kluczowa dla wyprowadzenia szybkich algorytmów z przerzedzeniem w czasie.

Przekształcenie kosinusowe drugiego rodzaju

Po wprowadzeniu pomocniczych ciągów $a(n)$ i $b(n)$, oraz po uwzględnieniu ich własności (4.20) i (4.21), wyrażenie (4.8), będące wzorem rozkładu szybkiego algorytmu dwuetapowego dla przekształcenia DCT-II postaci

$$X_N^{CII}(k) = \left[\sum_{n=0}^{N/2-1} (x(2n) + x(2n+1))\, cos\left(\frac{\pi}{(\frac{N}{2})} k(n+\frac{1}{2})\right)\right] cos\left(\frac{\pi}{2N}k\right) +$$

$$+ \left[\sum_{n=0}^{N/2-1} (-1)^n (x(2n) - x(2n+1))\, cos\left(\frac{\pi}{(\frac{N}{2})} (\frac{N}{2} - k)(n+\frac{1}{2})\right)\right] sin\left(\frac{\pi}{2N}k\right),$$

można zapisać jako

$$X_N^{CII}(k) = \left[\sum_{n=0}^{N/2-1} (a(n) + b(n))\, cos\left(\frac{\pi}{(\frac{N}{2})} k(n+\frac{1}{2})\right)\right] cos\left(\frac{\pi}{2N}k\right) +$$

$$+ \left[\sum_{n=0}^{N/2-1} (a(n) - b(n))\, cos\left(\frac{\pi}{(\frac{N}{2})} (\frac{N}{2} - k)(n+\frac{1}{2})\right)\right] sin\left(\frac{\pi}{2N}k\right).$$

Po rozpisaniu wyrażeń w nawiasach kwadratowych na osobne sumy otrzymujemy

$$\begin{aligned} X_N^{CII}(k) &= \left(\sum_{n=0}^{N/2-1} a(n) cos\left(\frac{\pi}{(\frac{N}{2})} k(n+\frac{1}{2})\right)\right) cos\left(\frac{\pi}{2N}k\right) + \\ &+ \left(\sum_{n=0}^{N/2-1} b(n) cos\left(\frac{\pi}{(\frac{N}{2})} k(n+\frac{1}{2})\right)\right) cos\left(\frac{\pi}{2N}k\right) + \\ &+ \left(\sum_{n=0}^{N/2-1} a(n) cos\left(\frac{\pi}{(\frac{N}{2})} \left(\frac{N}{2} - k\right)(n+\frac{1}{2})\right)\right) sin\left(\frac{\pi}{2N}k\right) - \\ &- \left(\sum_{n=0}^{N/2-1} b(n) cos\left(\frac{\pi}{(\frac{N}{2})} \left(\frac{N}{2} - k\right)(n+\frac{1}{2})\right)\right) sin\left(\frac{\pi}{2N}k\right). \end{aligned}$$

Sposób zapisu powyższej zależności można dodatkowo uprościć wprowadzając $N/2$ - punktowe przekształcenia DCT-II, określone odpowiednio na ciągach $a(n)$ i $b(n)$

$$X_2^{CII}(k) = \sum_{n=0}^{N/2-1} a(n) cos\left(\frac{\pi}{(\frac{N}{2})} k(n+\frac{1}{2})\right),$$

$$X_3^{CII}(k) = \sum_{n=0}^{N/2-1} b(n) cos\left(\frac{\pi}{(\frac{N}{2})} k(n+\frac{1}{2})\right).$$

Po ich podstawieniu otrzymujemy ostateczną postać wzoru rozkładu szybkiego algorytmu z przerzedzeniem w czasie dla dyskretnego przekształcenia kosinusowego drugiego rodzaju

$$\begin{aligned} X_N^{CII}(k) &= \left(X_2^{CII}(k) + X_3^{CII}(k)\right) cos\left(\frac{\pi}{2N}k\right) + \\ &+ \left(X_2^{CII}\left(\frac{N}{2}-k\right) - X_3^{CII}\left(\frac{N}{2}-k\right)\right) sin\left(\frac{\pi}{2N}k\right) \end{aligned}$$

dla $k = 0, 1, ..., N/2-1$,

$$\begin{aligned} X_N^{CII}(N-k) &= \left(X_2^{CII}\left(\frac{N}{2}-k\right) - X_3^{CII}\left(\frac{N}{2}-k\right)\right) cos\left(\frac{\pi}{2N}k\right) - \\ &- \left(X_2^{CII}(k) + X_3^{CII}(k)\right) sin\left(\frac{\pi}{2N}k\right) \end{aligned} \tag{4.22}$$

dla $k = 0, 1, ..., N/2-1$ oraz

$$X_N^{CII}(0) = X_2^{CII}(0) + X_3^{CII}(0),$$

$$X_N^{CII}(\tfrac{N}{2}) = \tfrac{\sqrt{2}}{2}(X_2^{CII}(0) - X_3^{CII}(0)).$$

Z zależności (4.22) wynika natychmiast, iż zgodnie ze wzorem rozkładu szybkiego algorytmu z przerzedzeniem w czasie, N - punktowe przekształcenie DCT-II obliczane jest na bazie dwóch $N/2$ - punktowych przekształceń tego samego typu, operujących bezpośrednio na elementach ciągów $a(n)$ i $b(n)$.

Należy tutaj zwrócić uwagę na fakt, iż ciągi $a(n)$ i $b(n)$ zawierają elementy wejściowego ciągu danych $x(n)$ o indeksach dobranych zgodnie z regułą (4.19). Zatem konkatenacja tych dwóch ciągów odpowiada przemieszanemu ciągowi wejściowemu $x(n)$. Stosując rekurencyjnie zależność (4.22) otrzymujemy algorytm pozwalający na szybkie obliczanie przekształcenia DCT-II o długości N będącej całkowitą potęgą dwóch. Mając na względzie schemat doboru elementów $x(n)$ dla ciągów $a(n)$ oraz $b(n)$, to dla algorytmu z przerzedzeniem w czasie otrzymujemy inne przemieszanie wejściowych elementów ciągu $x(n)$, niż $bit-reverse$ w przypadku szybkiego algorytmu dwuetapowego. W tabeli 4.3 pokazano sposób przyporządkowania indeksom n ich nowych wartości w zgodzie z kolejnością wymaganą dla algorytmu z przerzedzeniem w czasie (p. w cz.).

Na rysunku 4.2 pokazano graf przepływowy szybkiego algorytmu z przerzedzeniem w czasie dla przypadku $N = 16$ punktowego przekształcenia DCT-II. Z kolei w dodatku A można znaleźć odpowiednie procedury: realizującą permutację elementów $x(n)$

Tabela 4.3: Przyporządkowanie indeksów wg. kolejności z przerzedzeniem w czasie (p. w cz.) dla szesnastoelementowego ciągu danych

indeks	p. w cz.	indeks	p. w cz.
0	0	8	1
1	15	9	14
2	7	10	6
3	8	11	9
4	3	12	2
5	12	13	13
6	4	14	5
7	11	15	10

zgodnie z kolejnością charakterystyczną dla algorytmu z przerzedzeniem w czasie (patrz procedura A.3) oraz procedurę szybkiego obliczania DCT-II zgodnie ze wzorem (4.22) (patrz procedura A.4).

Ponieważ szybki algorytm obliczania przekształcenia DCT-II powstał na bazie szybkiego algorytmu dwuetapowego, jedynie poprzez zastosowanie innej niż $bit - reverse$ permutacji elementów wejściowego ciągu danych, to charakteryzuje go ta sama złożoność obliczeniowa. Stąd w danym przypadku dokładną liczbę operacji rzeczywistych dodawań i mnożeń również opisują zależności (4.11).

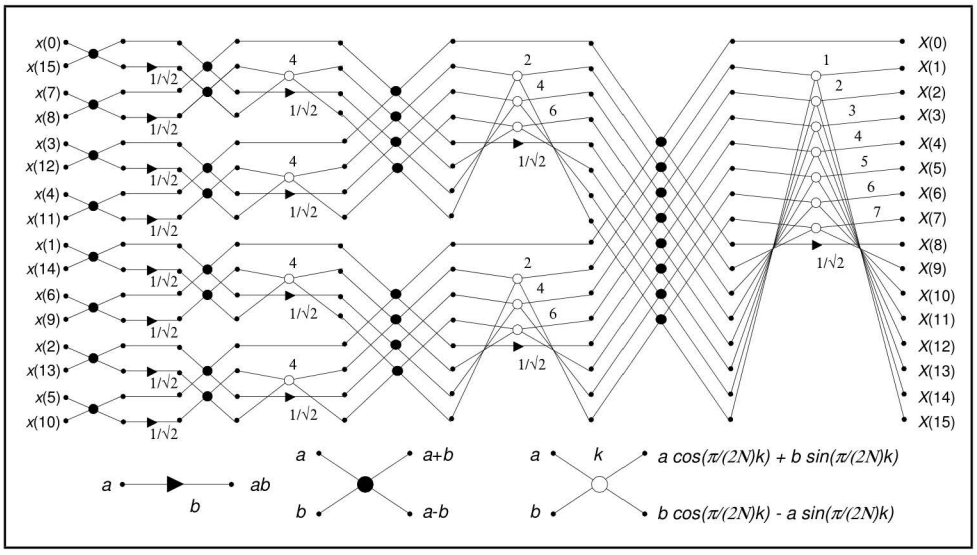
Przekształcenie kosinusowe czwartego rodzaju

W przypadku przekształcenia kosinusowego czwartego rodzaju postępujemy w sposób analogiczny, to znaczy wprowadzając ciągi zdefiniowane poprzez wyrażenia (4.19), otrzymujemy szybki algorytm z przerzedzeniem w czasie opisany następującym wzorem rozkładu

$$\begin{aligned} X_N^{CIV}(k) &= \left(X_2^{CIV}(k) + X_3^{CIV}(k)\right) cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) + \\ &+ \left(X_2^{CIV}(\frac{N}{2}-k-1) - X_3^{CIV}(\frac{N}{2}-k-1)\right) sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) \end{aligned} \tag{4.23}$$

oraz

$$\begin{aligned} X_N^{CIV}(N-k-1) &= \left(X_2^{CIV}(\frac{N}{2}-k-1) - X_3^{CIV}(\frac{N}{2}-k-1)\right) \\ &\quad cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) - \\ &- \left(X_2^{CIV}(k) + X_3^{CIV}(k)\right) sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) \end{aligned}$$



Rysunek 4.2: Graf przepływowy szybkiego algorytmu z przerzedzeniem w czasie dla $N = 16$ punktowego przekształcenia DCT-II

dla $k = 0, 1, ..., N/2 - 1$, gdzie

$$X_2^{CIV}(k) = \sum_{n=0}^{N/2-1} a(n) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right),$$

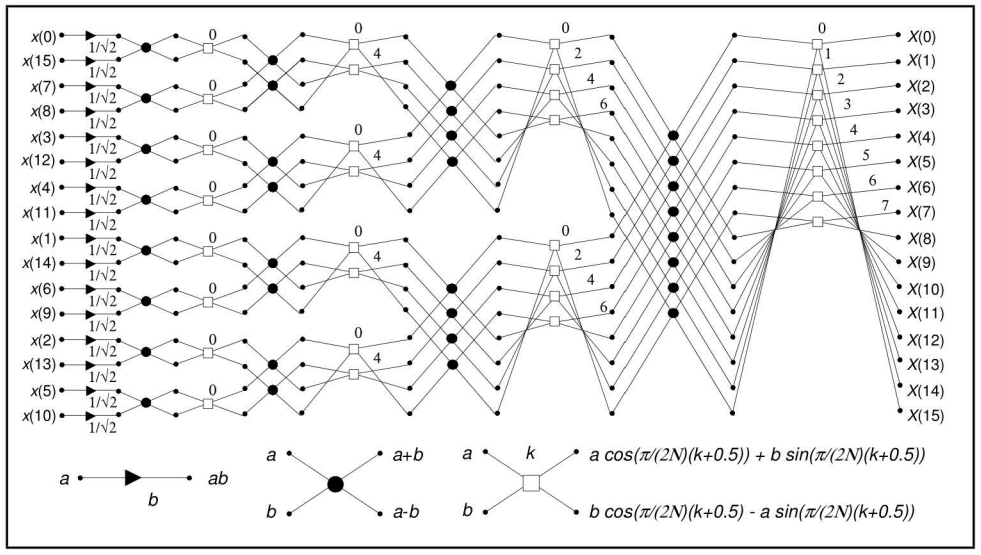
$$X_3^{CIV}(k) = \sum_{n=0}^{N/2-1} b(n) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$, to $N/2$ - punktowe przekształcenia DCT-IV operujące na ciągach $a(n)$ i $b(n)$. Tym samym otrzymujemy wzór rozkładu szybkiego algorytmu pozwalającego na obliczanie N-punktowego przekształcenia DCT-IV, na bazie dwóch $N/2$-punktowych przekształceń tego samego rodzaju, które operują bezpośrednio na elementach ciągu $x(n)$.

Poprzez rekurencyjne użycie powyższej zależności otrzymujemy szybki algorytm obliczania DCT-IV o długości będącej potęgą dwóch, dla którego liczby operacji rzeczywistych dodawań Do_N^{CIV} i mnożeń Mn_N^{CIV} opisują odpowiednio wzory (4.13). Graf przepływowy szybkiego algorytmu z przerzedzeniem w czasie dla $N = 16$ punktowego przekształcenia przedstawia rysunek 4.3. Procedura A.5, zamieszczona w dodatku A, zawiera natomiast implementację powyższego algorytmu w napisaną języku C.

Przekształcenie sinusowe drugiego rodzaju

W podobny sposób otrzymujemy wzory rozkładu szybkich algorytmów dla przekształceń sinusowych. Zaczynając od przekształcenia drugiego rodzaju i podstawiając wcześniej zdefiniowane ciągi $a(n)$ i $b(n)$ do wyrażenia (4.14), które opisuje szybki algorytm dwuetapowy, otrzymujemy wzór rozkładu szybkiego algorytmu z przerzedzeniem



Rysunek 4.3: Graf przepływowy szybkiego algorytmu z przerzedzeniem w czasie dla $N = 16$ punktowego przekształcenia DCT-IV

w czasie postaci

$$X_N^{SII}(k) = \left(X_2^{SII}(k) + X_3^{SII}(k)\right) cos\left(\frac{\pi}{2N}(k+1)\right) - \left(X_2^{SII}(\frac{N}{2}-k-2) - X_3^{SII}(\frac{N}{2}-k-2)\right) sin\left(\frac{\pi}{2N}(k+1)\right) \quad (4.24)$$

dla $k = 0, 1, ..., N/2-1$

$$X_N^{SII}(N-k-2) = \left(X_2^{SII}(k) + X_3^{SII}(k)\right) sin\left(\frac{\pi}{2N}(k+1)\right) + \left(X_2^{SII}(\frac{N}{2}-k-2) - X_3^{SII}(\frac{N}{2}-k-2)\right) cos\left(\frac{\pi}{2N}(k+1)\right)$$

dla $k = 0, 1, ..., N/2-1$ oraz

$$X_N^{SII}(\frac{N}{2}-1) = \left(X_2^{SII}(\frac{N}{2}-1) + X_3^{SII}(\frac{N}{2}-1)\right),$$

$$X_N^{SII}(N-1) = \frac{\sqrt{2}}{2}\left(X_2^{SII}(\frac{N}{2}-1) - X_3^{SII}(\frac{N}{2}-1)\right),$$

przy czym

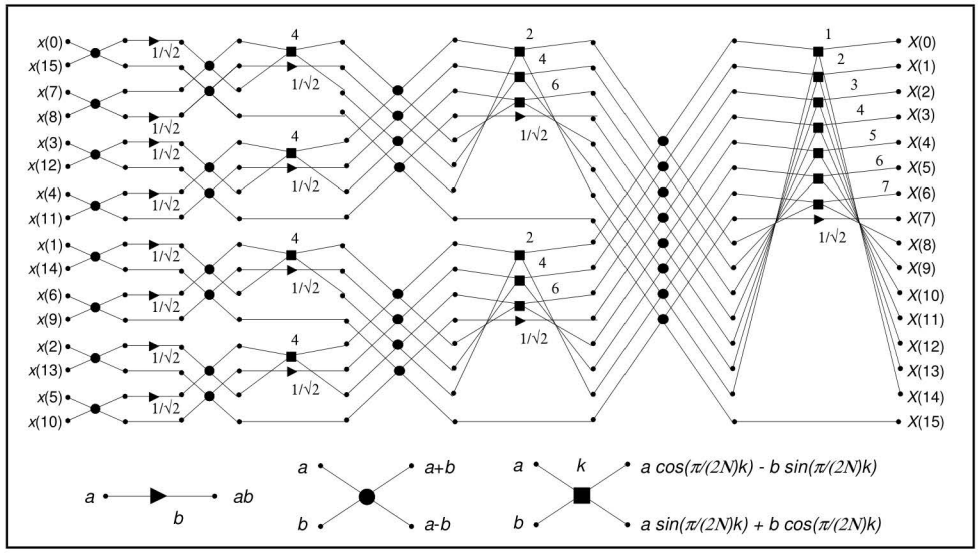
$$X_2^{SII}(k) = \sum_{n=0}^{N/2-1} a(n) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right),$$

$$X_3^{SII}(k) = \sum_{n=0}^{N/2-1} b(n) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$.

Rekurencyjne użycie powyższej zależności pozwala na szybkie obliczanie przekształcenia o długości N będącej potęgą dwóch, przy czym elementy wejściowego ciągu danych muszą być przemieszane zgodnie z kolejnością charakterystyczną dla proponowanych szybkich algorytmów z przerzedzeniem w czasie (patrz tabela 4.3). Graf przepływowy szybkiego algorytmu dla przypadku przekształcenia $N = 16$ punktowego przedstawiono na rysunku 4.4.

Złożoność obliczeniowa szybkiego algorytmu z przerzedzeniem w czasie jest identyczna jak algorytmu dwuetapowego. Zatem także i w tym przypadku dokładną liczbę operacji rzeczywistych dodawań Do_N^{SII} oraz mnożeń Mn_N^{SII} można wyznaczyć za pomocą wzorów (4.15).



Rysunek 4.4: Graf przepływowy szybkiego algorytmu z przerzedzeniem w czasie dla $N = 16$ punktowego przekształcenia DST-II

W dodatku A zamieszczono procedurę szybkiego obliczania DST-II z przerzedzeniem w czasie (patrz procedura A.6).

Przekształcenie sinusowe czwartego rodzaju

Ostatnim rozważanym przekształceniem jest przekształcenie sinusowe czwartego rodzaju. Także i tutaj, postępując w sposób analogiczny do pokazanego na przykładzie przekształcenia DCT-II, tzn. wprowadzając ciągi $a(n)$ i $b(n)$ oraz definiując pomocnicze przekształcenia $N/2$-punktowe postaci

$$X_2^{SIV}(k) = \sum_{n=0}^{N/2-1} a(n) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right),$$

$$X_3^{SIV}(k) = \sum_{n=0}^{N/2-1} b(n) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right),$$

otrzymujemy następujący wzór rozkładu szybkiego algorytmu dla tego przekształcenia

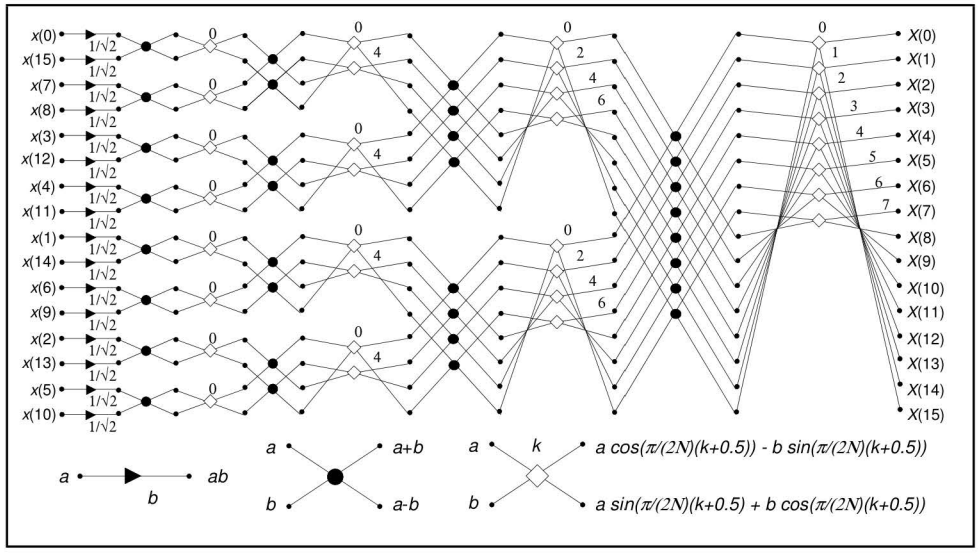
$$\begin{aligned} X_N^{SIV}(k) &= \left(X_2^{SIV}(k) + X_3^{SIV}(k)\right) cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) - \\ &- \left(X_2^{SIV}(\frac{N}{2}-k-1) - X_3^{SIV}(\frac{N}{2}-k-1)\right) sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) \end{aligned} \tag{4.25}$$

dla $k = 0, 1, ..., N/2 - 1$ oraz

$$\begin{aligned} X_N^{SIV}(N-k-1) &= \left(X_2^{SIV}(\frac{N}{2}-k-1) - X_3^{SIV}(\frac{N}{2}-k-1)\right) cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) + \\ &+ \left(X_2^{SIV}(k) + X_3^{SIV}(k)\right) sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right) \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 1$.

Graf szybkiego algorytmu z przerzedzeniem w czasie dla przypadku $N = 16$ punktowego przekształcenia DST-IV zamieszczono na rysunku 4.5.



Rysunek 4.5: Graf przepływowy szybkiego algorytmu z przerzedzeniem w czasie dla $N = 16$ punktowego przekształcenia DST-IV

Złożoność danego algorytmu jest identyczna jak algorytmu dwuetapowego. Stąd dokładną liczbę operacji rzeczywistych dodawań Do_N^{SIV} i mnożeń Mn_N^{SIV} można wyznaczyć za pomocą zależności (4.17).

W dodatku A zamieszczono procedurę obliczania szybkiego algorytmu z przerzedzeniem w czasie dla DST-IV (patrz procedura A.7).

W niniejszej sekcji przedstawione zostały wzory rozkładu szybkich algorytmów dwuetapowych, oraz szybkich algorytmów z przerzedzeniem w czasie dla dyskretnych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju. Ponadto zamieszczono grafy przepływowe dla szybkich algorytmów z przerzedzeniem w czasie dla

przypadków, gdy liczba próbek N wynosiła 16. Dodatkowo do każdego z przekształceń dołączono procedury napisane w języku C (patrz dodatek A), które zawierają ich implementacje.

Proponowane szybkie algorytmy otrzymano na podstawie znanych algorytmów dwuetapowych, jedynie poprzez wprowadzenie nowej permutacji elementów wejściowego ciągu danych. Stąd charakteryzuje je ta sama złożoność obliczeniowa co znane algorytmy dwuetapowe. Ponadto, jak pokazano, algorytmy z przerzedzeniem w czasie posiadają struktury odpowiednie dla szybkiego obliczania przekształceń całkowych według proponowanych schematów adaptacyjnych, które przedstawiono w rozdziale 3.

Dalszą część niniejszego rozdziału wypełnia przypadek przekształceń dwuwymiarowych, które przez wzgląd na specyfikę algorytmów adaptacyjnych, nie mogą być obliczane według znanego podejścia wierszowo-kolumnowego.

4.2. Szybkie algorytmy dla dyskretnych przekształceń dwuwymiarowych

Dwuwymiarowe całkowe przekształcenie Fouriera, a także przekształcenia kosinusowe i sinusowe Fouriera, należą do klasy przekształceń o separowalnych funkcjach bazowych. Stąd w praktyce szybkie algorytmy dla dyskretnych dwuwymiarowych przekształceń Fouriera oraz kosinusowych i sinusowych przekształceń drugiego i czwartego rodzaju, bardzo często buduje się według tzw. schematu wierszowo-kolumnowego, z wykorzystaniem szybkich algorytmów dla przekształceń jednowymiarowych. Podejście to polega na stosowaniu przekształcenia jednowymiarowego początkowo do wierszy (lub kolumn), a następnie do kolumn (lub wierszy) macierzy próbek sygnału wejściowego.

Jak już wspomniano, do budowy szybkich algorytmów adaptacyjnego obliczania wskazanych przekształceń dyskretnych potrzeba szybkich algorytmów, które umożliwią obliczanie przekształcenia $2N$ na $2M$ - punktowego, będącego odpowiednikiem kubatur wyznaczanych na danym etapie adaptacji, bezpośrednio na bazie przekształcenia N na M-punktowego, które stanowi zespół kubatur liczonych na etapie poprzedzającym (patrz rozdział 3). Stąd w danym przypadku nie jest możliwe zastosowanie schematu wierszowo-kolumnowego. Powyższe wymagania spełniają natomiast szybkie algorytmy z przerzedzeniem w czasie, które przedstawione zostaną w dalszej części niniejszego rozdziału.

W sekcji 4.2.1 zamieszczono wzór rozkładu szybkiego algorytmu typu radix-2 dla dyskretnego dwuwymiarowego przekształcenia Fouriera (DFT2D) [8]. Z kolei w sekcji 4.2.2 przedstawiono wzory rozkładów szybkich algorytmów dwuetapowych dla dyskretnych dwuwymiarowych przekształceń kosinusowych i sinusowych, a następnie na ich podstawie, poprzez wprowadzenie specjalnej permutacji elementów wejściowej macierzy danych, zbudowano szukane szybkie algorytmy z przerzedzeniem w czasie (patrz sekcja 4.2.3). Przez wzgląd na znaczne analogie obecne we wszystkich rozważanych przypadkach, pełne wyprowadzenia wzorów rozkładu szybkich algorytmów pokazano jedynie na przykładzie dwuwymiarowego dyskretnego przekształcenia kosinusowego drugiego rodzaju (DCT2D-II).

W dodatku A zamieszczono procedury realizujące opracowane szybkie algorytmy z przerzedzeniem w czasie. Ponadto do opisu każdego z przedstawionych szybkich algorytmów załączono wyrażenia pozwalające na dokładne wyznaczenie liczby operacji arytmetycznych dodawań i mnożeń.

4.2.1. Szybki algorytm dla dyskretnego dwuwymiarowego przekształcenia Fouriera

Niech $x(n,m)$ dla $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$ oznacza N na M elementową macierz danych wejściowych. Wówczas zgodnie z rozdziałem 2 dyskretne dwuwymiarowe przekształcenie Fouriera definiuje się jako

$$X^{2D}_{N\cdot M}(k,l) = DFT2D_{N\cdot M}\left\{x(n,m)\right\} \triangleq \frac{1}{NM}\left[\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m)W_N^{kn}W_M^{lm}\right] \tag{4.26}$$

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

$$W_N^{kn} = e^{-i\frac{2\pi}{N}kn} = cos\left(\frac{2\pi}{N}kn\right) - isin\left(\frac{2\pi}{N}kn\right),$$

$$W_N^{lm} = e^{-i\frac{2\pi}{M}lm} = cos\left(\frac{2\pi}{M}lm\right) - isin\left(\frac{2\pi}{M}lm\right).$$

Pomijając współczynnik normalizujący $\frac{1}{NM}$ i rozpisując każdą z sum z powyższego wyrażenia na dwie sumy elementów o indeksach parzystych i nieparzystych, a także odpowiednio grupując czynniki, otrzymujemy wzór rozkładu szybkiego algorytmu z przerzedzeniem w czasie dla dwuwymiarowego przekształcenia Fouriera (FFT2D). Wzór ten opisuje poniższa zależność [8]

$$\begin{aligned} X^{2D}_{N\cdot M}(k,l) &= X_0^{2D}(k,l) + X_1^{2D}(k,l)W_N^k + \\ &+ X_2^{2D}(k,l)W_M^l + X_3^{2D}(k,l)W_N^kW_M^l \end{aligned} \tag{4.27}$$

dla $k = 0, 1, ..., N/2-1$, $l = 0, 1, ..., M/2-1$,

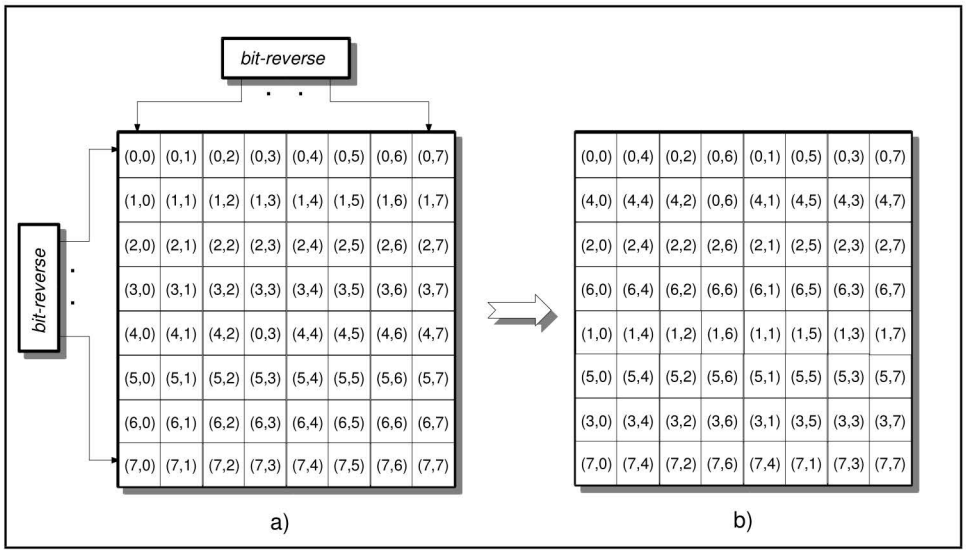
$$\begin{aligned} X^{2D}_{N\cdot M}(k+\frac{N}{2},l) &= X_0^{2D}(k,l) - X_1^{2D}(k,l)W_N^k + \\ &+ X_2^{2D}(k,l)W_M^l - X_3^{2D}(k,l)W_N^kW_M^l \end{aligned}$$

dla $k = 0, 1, ..., N/2-1$, $l = 0, 1, ..., M/2-1$,

$$\begin{aligned} X^{2D}_{N\cdot M}(k,l+\frac{M}{2}) &= X_0^{2D}(k,l) + X_1^{2D}(k,l)W_N^k - \\ &- X_2^{2D}(k,l)W_M^l - X_3^{2D}(k,l)W_N^kW_M^l \end{aligned}$$

dla $k = 0, 1, ..., N/2-1$, $l = 0, 1, ..., M/2-1$,

$$\begin{aligned} X^{2D}_{N\cdot M}(k+\frac{N}{2},l+\frac{M}{2}) &= X_0^{2D}(k,l) - X_1^{2D}(k,l)W_N^k - \\ &- X_2^{2D}(k,l)W_M^l + X_3^{2D}(k,l)W_N^kW_M^l \end{aligned}$$



Rysunek 4.6: Przyporządkowanie elementom macierzy danych wejściowych (a) nowych indeksów wg. dwuwymiarowej permutacji $bit-reverse$ (b) dla przypadku $N = M = 8$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$, gdzie

$$X_0^{2D}(k,l) = DFT2D_{\frac{N}{2}\cdot\frac{M}{2}}\{x(2n,2m)\},$$

$$X_1^{2D}(k,l) = DFT2D_{\frac{N}{2}\cdot\frac{M}{2}}\{x(2n+1,2m)\},$$

$$X_2^{2D}(k,l) = DFT2D_{\frac{N}{2}\cdot\frac{M}{2}}\{x(2n,2m+1)\},$$

$$X_3^{2D}(k,l) = DFT2D_{\frac{N}{2}\cdot\frac{M}{2}}\{x(2n+1,2m+1)\}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$. Z wyrażenia (4.27) wynika, iż według wzoru rozkładu szybkiego algorytmu z przerzedzeniem w czasie typu radix-2 przekształcenie N na M-punktowe obliczanie jest na podstawie czterech przekształceń $N/2$ na $M/2$-punktowych, które operują na elementach $x(2n, 2m)$, $x(2n+1, 2m)$, $x(2n, 2m+1)$ i $x(2n+1, 2m+1)$ dla $n = 0, 1, ..., N/2 - 1$ i $m = 0, 1, ..., M/2 - 1$. Przekształcenia te wymnaża się dodatkowo poprzez następujące współczynniki zespolone W_N^k oraz W_M^l. Rekurencyjne zastosowanie wzoru (4.27) umożliwia szybkie obliczanie przekształcenia dwuwymiarowego o długościach N i M będących całkowitymi potęgami liczby dwa. Taka operacja wymusza jednak przemieszanie elementów wejściowej macierzy $x(n, m)$ zgodnie z kolejnością $bit - reverse$, zastosowaną odpowiednio do jej wierszy i kolumn. Na rysunku 4.6 pokazano sposób przyporządkowania elementom macierzy $x(n, m)$ nowych indeksów dla przypadku $N = M = 8$ punktów.

W dodatku A zamieszczono procedurę realizującą wymagane przemieszanie (patrz procedura A.8), a także kod źródłowy procedury obliczania dyskretnego dwuwymiarowego przekształcenia Fouriera, realizowanego według struktury szybkiego algorytmu z przerzedzeniem w czasie (patrz procedura A.9).

Złożoność obliczeniowa N na M-punktowego przekształcenia DFT2D, liczonego bezpośrednio z definicji (4.26), jest rzędu $\mathcal{O}(N^4)$. Zastosowanie szybkiego algorytmu redukuje tę złożoność do poziomu $\mathcal{O}(N^2 log_2 N)$. Poniższe zależności pozwalają na dokładne wyznaczenie liczby rzeczywistych operacji dodawań $Dod^{2D}_{N\cdot M}$

$$Dod^{2D}_{N\cdot M} = \begin{cases} 3MN\left(\frac{5}{2}\log_2 M + \log_2 K - 1\right) + \frac{1}{3}M^2 + M + N + \frac{2}{3} & \text{dla } N \geqslant M, \\ 3MN\left(\frac{5}{2}\log_2 N - \log_2 K - 1\right) + \frac{1}{3}N^2 + N + M + \frac{2}{3} & \text{dla } N < M \end{cases}$$

i mnożeń $Mn^{2D}_{N\cdot M}$

$$Mn^{2D}_{N\cdot M} = \begin{cases} 2MN\left(\frac{3}{2}\log_2 M + \log_2 K - 3\right) + \frac{2}{3}M^2 + 2(M+N) + \frac{4}{3} & \text{dla } N \geqslant M, \\ 2MN\left(\frac{3}{2}\log_2 N - \log_2 K - 3\right) + \frac{2}{3}N^2 + 2(N+M) + \frac{4}{3} & \text{dla } N < M, \end{cases}$$

gdzie $K = N/M$.

4.2.2. Szybkie algorytmy dwuetapowe dla dyskretnych dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju

Za punkt wyjścia do budowy szybkich algorytmów z przerzedzeniem w czasie dla dyskretnych dwuwymiarowych przekształceń: kosinusowego drugiego rodzaju (DCT2D-II), sinusowego drugiego rodzaju (DST2D-II), kosinusowego (DCT2D-IV) oraz sinusowego czwartego rodzaju (DST2D-IV) przyjęto struktury znanych algorytmów dwuetapowych [156], które rozszerzono na przypadki przekształceń dwuwymiarowych. Na ich podstawie, poprzez wprowadzenie odpowiedniego przemieszania elementów wejściowej macierzy danych $x(n,m)$, uzyskano wymagane szybkie algorytmy z przerzedzeniem w czasie. W niniejszej sekcji przedstawione zostały wzory rozkładów algorytmów dwuetapowych, przy czym dokładne ich wyprowadzenia pokazano jedynie dla przypadku przekształcenia DCT2D-II.

Dwuwymiarowe dyskretne przekształcenie kosinusowe drugiego rodzaju

Niech $x(n,m)$ oznacza dwuwymiarową tablicę o N wierszach i M kolumnach. Wówczas dyskretne dwuwymiarowe przekształcenie kosinusowe drugiego rodzaju, określone na $x(n,m)$, zgodnie z rozdziałem 2, definiuje zależność

$$\begin{aligned} X^{C2D(II)}_{N\cdot M}(k,l) &= DCT2DII\{x(n,m)\} = \\ &= \frac{p_k p_l}{NM}\left[\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m) cos\left(\frac{\pi}{N}k(n+\frac{1}{2})\right) cos\left(\frac{\pi}{M}l(m+\frac{1}{2})\right)\right] \end{aligned}$$

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

$$p_s = \begin{cases} 2 \text{ dla } s = 0, \\ 1 \text{ dla } s \neq 0. \end{cases} \tag{4.28}$$

W dalszych rozważaniach pominięto współczynniki $\frac{4}{NM}$, p_k oraz p_l, przez które mnożenia można wykonać na samym końcu. Wówczas rozpisując sumy w powyższym wyrażeniu na sumy po parzystych i nieparzystych indeksach n i m, otrzymujemy wzór postaci

$$\begin{aligned} X_{N\cdot M}^{C2DII}(k,l) &= \sum_{n=0}^{N/2-1}\sum_{m=0}^{M-1} x(2n,m)cos\left(\frac{\pi}{N}k(2n+\frac{1}{2})\right)cos\left(\frac{\pi}{M}l(m+\frac{1}{2})\right)+ \\ &+ \sum_{n=0}^{N/2-1}\sum_{m=0}^{M-1} x(2n+1,m)cos\left(\frac{\pi}{N}k(2n+1+\frac{1}{2})\right)cos\left(\frac{\pi}{M}l(m+\frac{1}{2})\right)= \\ &= \sum_{n=0}^{N/2-1}\sum_{m=0}^{M-1} x(2n,m)cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})-\frac{\pi}{2N}k\right)cos\left(\frac{\pi}{M}l(m+\frac{1}{2})\right)+ \\ &+ \sum_{n=0}^{N/2-1}\sum_{m=0}^{M-1} x(2n+1,m)cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})+\frac{\pi}{2N}k\right) \\ &\quad \cdot cos\left(\frac{\pi}{M}l(m+\frac{1}{2})\right). \end{aligned} \tag{4.29}$$

Korzystając z własności kosinusa sumy kątów: $cos(\alpha+\beta)=cos(\alpha)cos(\beta)-sin(\alpha)sin(\beta)$ oraz różnicy kątów: $cos(\alpha-\beta)=cos(\alpha)cos(\beta)+sin(\alpha)sin(\beta)$, a także mając na uwadze Własność (2.4.1), to wyrażenie (4.29) można przekształcić do następującej postaci

$$\begin{aligned} X_{N\cdot M}^{C2DII}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M-1}&\left(x(2n,m)+x(2n+1,m)\right)cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})\right) \\ &\cdot cos\left(\frac{\pi}{2N}k\right)cos\left(\frac{\pi}{M}l(m+\frac{1}{2})\right)+ \\ +\sum_{n=0}^{N/2-1}\sum_{m=0}^{M-1}&(-1)^n\left(x(2n,m)-x(2n+1,m)\right)cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(\frac{N}{2}-k)(n+\frac{1}{2})\right) \\ &\cdot sin\left(\frac{\pi}{2N}k\right)cos\left(\frac{\pi}{M}l(m+\frac{1}{2})\right). \end{aligned}$$

Postępując w sposób analogiczny względem indeksu m, a także wprowadzając oparte na kombinacjach elementów $x(n,m)$ pomocnicze przekształcenia DCT2D-II postaci

$$\begin{aligned} X_0^{C2DII}(k,l) &= \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1}\left(x(2n,2m)+x(2n+1,2m)+x(2n,2m+1)+\right. \\ &+ \left. x(2n+1,2m+1)\right)cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}k(n+\frac{1}{2})\right)cos\left(\frac{\pi}{\left(\frac{M}{2}\right)}l(m+\frac{1}{2})\right) \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$,

$$X_1^{C2DII}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} (-1)^n (x(2n,2m) - x(2n+1,2m) + x(2n,2m+1) + \\ - x(2n+1,2m+1)) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)} k(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)} l(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$,

$$X_2^{C2DII}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} (-1)^m (x(2n,2m) + x(2n+1,2m) - x(2n,2m+1) - \\ - x(2n+1,2m+1)) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)} k(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)} l(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$, oraz

$$X_3^{C2DII}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} (-1)^{m+n} (x(2n,2m) - x(2n+1,2m) - x(2n,2m+1) + \\ + x(2n+1,2m+1)) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)} k(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)} l(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$, wraz z następującymi symbolami pomocniczymi dla skrócenia zapisu

$$C_N^k = cos\left(\frac{\pi}{2N} k\right), \; S_N^k = sin\left(\frac{\pi}{2N} k\right), \\ C_M^l = cos\left(\frac{\pi}{2M} l\right), \; S_M^l = sin\left(\frac{\pi}{2M} l\right), \tag{4.30}$$

to wzór rozkładu szybkiego algorytmu z przerzedzeniem w czasie dla dyskretnego przekształcenia kosinusowego drugiego rodzaju (FCT2D-II), można zapisać w ostatecznej postaci jako

$$X_{N\cdot M}^{C2DII}(k,l) = X_0^{C2DII}(k,l) C_N^k C_M^l + X_1^{C2DII}(\frac{N}{2} - k, l) S_N^k C_M^l + \\ + X_2^{C2DII}(k, \frac{M}{2} - l) C_N^k S_M^l + X_3^{C2DII}(\frac{N}{2} - k, \frac{M}{2} - l) S_N^k S_M^l \tag{4.31}$$

dla $k = 1, 2, ..., N/2 - 1$ oraz $l = 1, 2, ..., M/2 - 1$. Wyrażenie to pozwala na syntezę szybkiego dwuetapowego algorytmu dla przekształcenia DCT2D-II. Jednak dla potrzeb implementacyjnych wygodne jest rozpisanie powyższej zależności z uwzględnieniem wszystkich przypadków szczególnych, wynikających z symetrii dwuwymiarowego dyskretnego przekształcenia kosinusowego drugiego rodzaju. Tak rozpisaną postać wzoru (4.31) zamieszczono poniżej drobnym drukiem. Postać tę uzyskano na podstawie

Własności (2.5.2) - (2.5.3) oraz (2.5.4) - (2.5.6) przedstawionych w rozdziale 2 niniejszej monografii.

$$\begin{aligned} X_{N\cdot M}^{C2DII}(N-k,l) &= -X_0^{C2DII}(k,l)S_N^k C_M^l + X_1^{C2DII}(\tfrac{N}{2}-k,l)C_N^k C_M^l - \\ &- X_2^{C2DII}(k,\tfrac{M}{2}-l)S_N^k S_M^l + X_3^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l)C_N^k S_M^l \end{aligned}$$

dla $k = 1, 2, ..., N/2-1$, $l = 1, 2, ..., M/2-1$,

$$\begin{aligned} X_{N\cdot M}^{C2DII}(k,M-l) &= -X_0^{C2DII}(k,l)C_N^k S_M^l - X_1^{C2DII}(\tfrac{N}{2}-k,l)S_N^k S_M^l + \\ &+ X_2^{C2DII}(k,\tfrac{M}{2}-l)C_N^k C_M^l + X_3^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l)S_N^k C_M^l \end{aligned}$$

dla $k = 1, 2, ..., N/2-1$, $l = 1, 2, ..., M/2-1$,

$$\begin{aligned} X_{N\cdot M}^{C2DII}(N-k,M-l) &= X_0^{C2DII}(k,l)S_N^k S_M^l - X_1^{C2DII}(\tfrac{N}{2}-k,l)C_N^k S_M^l - \\ &- X_2^{C2DII}(k,\tfrac{M}{2}-l)S_N^k C_M^l + \\ &+ X_3^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l)C_N^k C_M^l \end{aligned}$$

dla $k = 1, 2, ..., N/2-1$, $l = 1, 2, ..., M/2-1$,

$$X_{N\cdot M}^{C2DII}(0,0) = X_0^{C2DII}(0,0),\ X_{N\cdot M}^{C2DII}(\tfrac{N}{2},0) = \tfrac{\sqrt{2}}{2}X_1^{C2DII}(0,0),$$
$$X_{N\cdot M}^{C2DII}(0,\tfrac{M}{2}) = \tfrac{\sqrt{2}}{2}X_2^{C2DII}(0,0),\ X_{N\cdot M}^{C2DII}(\tfrac{N}{2},\tfrac{M}{2}) = \tfrac{1}{2}X_3^{C2DII}(0,0)$$

oraz

$$\begin{aligned} X_{N\cdot M}^{C2DII}(k,0) &= X_0^{C2DII}(k,0)C_N^k + X_1^{C2DII}(\tfrac{N}{2}-k,0)S_N^k, \\ X_{N\cdot M}^{C2DII}(N-k,0) &= -X_0^{C2DII}(k,0)S_N^k + X_1^{C2DII}(\tfrac{N}{2}-k,0)C_N^k, \\ X_{N\cdot M}^{C2DII}(k,\tfrac{M}{2}) &= \tfrac{\sqrt{2}}{2}X_2^{C2DII}(k,0)C_N^k + \tfrac{\sqrt{2}}{2}X_3^{C2DII}(\tfrac{N}{2}-k,0)S_N^k, \\ X_{N\cdot M}^{C2DII}(N-k,\tfrac{M}{2}) &= -\tfrac{\sqrt{2}}{2}X_2^{C2DII}(k,0)S_N^k + \tfrac{\sqrt{2}}{2}X_3^{C2DII}(\tfrac{N}{2}-k,0)C_N^k \end{aligned}$$

dla $k = 1, 2, ..., N/2-1$, a także

$$\begin{aligned} X_{N\cdot M}^{C2DII}(0,l) &= X_0^{C2DII}(0,l)C_N^k + X_2^{C2DII}(0,\tfrac{M}{2}-l)S_M^l, \\ X_{N\cdot M}^{C2DII}(0,M-l) &= -X_0^{C2DII}(0,l)S_N^k + X_2^{C2DII}(0,\tfrac{M}{2}-l)C_M^l, \\ X_{N\cdot M}^{C2DII}(\tfrac{N}{2},l) &= \tfrac{\sqrt{2}}{2}X_1^{C2DII}(0,l)C_N^k + \tfrac{\sqrt{2}}{2}X_3^{C2DII}(0,\tfrac{M}{2}-l)S_M^l, \\ X_{N\cdot M}^{C2DII}(\tfrac{N}{2},M-l) &= -\tfrac{\sqrt{2}}{2}X_1^{C2DII}(0,l)S_N^k + \tfrac{\sqrt{2}}{2}X_3^{C2DII}(0,\tfrac{M}{2}-l)C_M^l \end{aligned}$$

dla $l = 1, 2, ..., M/2-1$.

Z zależności (4.31) wynika, iż dla szybkiego obliczenia dwuwymiarowego przekształcenia DCT2D-II o N na M punktach, potrzeba czterech przekształceń $N/2$ na $M/2$-punktowych tego samego rodzaju, które operują na pewnych kombinacjach elementów macierzy $x(n,m)$, a także prostych operacji arytmetycznych w postaci rzeczywistych dodawań i mnożeń przez stałe opisane zależnością (4.30). Zatem konstrukcja algorytmu wyklucza jego użycie do szybkiego obliczania dwuwymiarowego przekształcenia kosinusowego według zaproponowanego w rozdziale 3 schematu adaptacyjnego.

Rekurencyjne użycie wzoru (4.31) umożliwia konstrukcję szybkiego algorytmu obliczania dwuwymiarowego przekształcenia kosinusowego drugiego rodzaju o długości N na M punktów, gdzie N i M są całkowitymi potęgami dwóch. Szybki algorytm dwuetapowy zbudowany zgodnie z przedstawionym wzorem redukuje złożoność obliczeniową DCT2D-II z poziomu $\mathcal{O}(N^4)$ do poziomu $\mathcal{O}(N^2 log_2 N)$. Elementy wejściowej tablicy $x(n,m)$ muszą być posortowane zgodnie z kolejnością $bit - reverse$, którą stosuje się najpierw do wierszy (kolumn), a następnie kolumn (wierszy) macierzy $x(n,m)$.

Na dokładne wyznaczenie liczby operacji rzeczywistych dodawań $Do_{N\cdot M}^{C2DII}$ oraz mnożeń $Mn_{N\cdot M}^{C2DII}$ pozwalają poniższe zależności

$$Do_{N\cdot M}^{C2DII} = \begin{cases} 6MN\left(\log_2 M + \frac{1}{3}\log_2 K - 1\right) - \frac{2}{3}M^2 + 4(M+N) - \frac{4}{3} & \text{dla } N \geqslant M, \\ 6MN\left(\log_2 N - \frac{1}{3}\log_2 K - 1\right) - \frac{2}{3}N^2 + 4(M+N) - \frac{4}{3} & \text{dla } N < M \end{cases}$$

oraz

$$Mn_{N\cdot M}^{C2DII} = \begin{cases} 8MN\left(\log_2 M + \frac{1}{4}\log_2 K - \frac{7}{4}\right) - 3M^2 + 11(M+N) - 5 & \text{dla } N \geqslant M, \\ 8MN\left(\log_2 N - \frac{1}{4}\log_2 K - \frac{7}{4}\right) - 3N^2 + 11(M+N) - 5 & \text{dla } N < M, \end{cases}$$

gdzie $K = N/M$.

W analogiczny sposób buduje się szybkie algorytmy obliczania dwuwymiarowych dyskretnych przekształceń kosinusowego czwartego rodzaju oraz sinusowego drugiego i czwartego rodzaju. Wzory rozkładu dla tych algorytmów przedstawione zostały w dalszej części niniejszego rozdziału.

Dwuwymiarowe dyskretne przekształcenie kosinusowe czwartego rodzaju

Dyskretne dwuwymiarowe przekształcenie kosinusowe czwartego (DCT2D-IV) rodzaju definiuje następujące wyrażenie

$$X_{N\cdot M}^{C2DIV}(k,l) = DCT2DIV_{N\cdot M}\{x(n,m)\} \triangleq$$

$$\triangleq \frac{1}{NM}\left[\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m) cos\left(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2})\right) cos\left(\frac{\pi}{M}(l+\frac{1}{2})(m+\frac{1}{2})\right)\right],$$

gdzie $k = 0, 1, ..., N-1$ i $l = 0, 1, ..., M-1$. Pomijając czynnik normalizujący $\frac{1}{MN}$, oraz stosując analogiczne przekształcenia do tych, których użyto podczas wyprowadzania dwuetapowego algorytmu FCT2D-II, otrzymujemy wzór rozkładu szybkiego algorytmu dwuetapowego dla danego przekształcenia.

Wprowadźmy następujące $N/2$ na $M/2$-punktowe przekształcenia pomocnicze, operujące na kombinajach elementów $x(n,m)$ o indeksach parzystych i nieparzystych

$$X_0^{C2DIV}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (x(2n,2m) + x(2n+1,2m) + x(2n,2m+1) +$$

$$+x(2n+1,2m+1)) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$,

$$X_1^{C2DIV}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} (-1)^n (x(2n, 2m) - x(2n+1, 2m) + x(2n, 2m+1) +$$

$$-x(2n+1, 2m+1)))cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$,

$$X_2^{C2DIV}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} (-1)^m (x(2n, 2m) + x(2n+1, 2m) - x(2n, 2m+1) -$$

$$-x(2n+1, 2m+1))cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$,

$$X_3^{C2DIV}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} (-1)^{m+n} (x(2n, 2m) - x(2n+1, 2m) - x(2n, 2m+1) +$$

$$+x(2n+1, 2m+1))cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$, wraz z następującymi symbolami wymaganymi dla skrócenia zapisu

$$C_N^{(k+\frac{1}{2})} = cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right),\ S_N^{(k+\frac{1}{2})} = sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right),$$

$$C_M^{(l+\frac{1}{2})} = cos\left(\frac{\pi}{2M}(l+\frac{1}{2})\right),\ S_M^{(l+\frac{1}{2})} = sin\left(\frac{\pi}{2M}(l+\frac{1}{2})\right).$$

Wówczas wzór rozkładu szybkiego algorytmu dwuetapowego (FCT2D-IV) dla przekształcenia DCT2D-IV można zapisać w postaci

$$\begin{aligned} X_{N\cdot M}^{C2DIV}(k,l) &= X_0^{C2DIV}(k,l) C_N^{(k+\frac{1}{2})} C_M^{(l+\frac{1}{2})} + \\ &+ X_1^{C2DIV}(\frac{N}{2} - k - 1, l) S_N^{(k+\frac{1}{2})} C_M^{(l+\frac{1}{2})} + X_2^{C2DIV}(k, \frac{M}{2} - l - 1) C_N^{(k+\frac{1}{2})} S_M^{(l+\frac{1}{2})} + \\ &+ X_3^{C2DIV}(\frac{N}{2} - k - 1, \frac{M}{2} - l - 1) S_N^{(k+\frac{1}{2})} S_M^{(l+\frac{1}{2})} \end{aligned} \tag{4.32}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$. Powyższy wzór może stanowić podstawę dla implementacji algorytmu FCT2D-IV. Wygodniejszą postać, która uwzględnia własności symetrii przekształceń składowych wzoru (4.32), zamieszczono poniżej drobnym drukiem.

$$\begin{aligned} X_{N\cdot M}^{C2DIV}(N-k-1,l) &= -X_0^{C2DIV}(k,l)S_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})}+ \\ &+ X_1^{C2DIV}(\tfrac{N}{2}-k-1,l)C_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})}-X_2^{C2DIV}(k,\tfrac{M}{2}-l-1)S_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})}+ \\ &+ X_3^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)C_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} \end{aligned}$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$,

$$\begin{aligned} X_{N\cdot M}^{C2DIV}(k,M-l-1) &= -X_0^{C2DIV}(k,l)C_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})}- \\ &- X_1^{C2DIV}(\tfrac{N}{2}-k-1,l)S_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})}+X_2^{C2DIV}(k,\tfrac{M}{2}-l-1)C_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})}+ \\ &+ X_3^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)S_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} \end{aligned}$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$ oraz

$$\begin{aligned} X_{N\cdot M}^{C2DIV}(N-k-1,M-l-1) &= X_0^{C2DIV}(k,l)S_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})}- \\ &- X_1^{C2DIV}(\tfrac{N}{2}-k-1,l)C_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})}-X_2^{C2DIV}(k,\tfrac{M}{2}-l-1)S_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})}+ \\ &+ X_3^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)C_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} \end{aligned}$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$.

Zgodnie z wzorem (4.32) obliczanie dwuwymiarowego N na M-punktowego przekształcenia DCT2D-IV odbywa się na podstawie czterech przekształceń $N/2$ na $M/2$-punktowych, które operują na kombinacjach elementów macierzy $x(n,m)$, przy czym wiersze i kolumny tej macierzy muszą być przemieszane zgodnie z kolejnością $bit-reverse$.

Złożoność obliczeniowa przedstawionego dwuetapowego algorytmu FCT2D-IV jest rzędu $\mathcal{O}(N^2 log_2 N)$. Dokładną liczbę operacji rzeczywistych dodawań $Do_{N\cdot M}^{C2DIV}$ oraz mnożeń $Mn_{N\cdot M}^{C2DIV}$ podają poniższe zależności

$$Do_{N\cdot M}^{C2DIV} = \begin{cases} 6MN\left(\log_2 M + \frac{1}{3}\log_2 K\right) & \text{dla } N \geqslant M, \\ 6MN\left(\log_2 N - \frac{1}{3}\log_2 K\right) & \text{dla } N < M \end{cases}$$

oraz

$$Mn_{N\cdot M}^{C2DIV} = \begin{cases} 8MN\left(\log_2 M + \frac{1}{4}\log_2 K + \frac{1}{8}\right) & \text{dla } N \geqslant M, \\ 8MN\left(\log_2 N - \frac{1}{4}\log_2 K + \frac{1}{8}\right) & \text{dla } N < M, \end{cases}$$

gdzie $K = N/M$.

Dwuwymiarowe dyskretne przekształcenie sinusowe drugiego rodzaju

Niech $x(n,m)$ będzie macierzą o N wierszach i M kolumnach. Wówczas dyskretne dwuwymiarowe przekształcenie sinusowe drugiego rodzaju (DST2D-II), określone na

elementach tablicy $x(n,m)$, definiuje się jako

$$X_{N\cdot M}^{S2DII}(k,l) = DST2DII_{N\cdot M}\{x(n,m)\} \triangleq$$

$$\triangleq \frac{p_k p_l}{NM}\left[\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m) sin\left(\frac{\pi}{N}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{M}(l+1)(m+\frac{1}{2})\right)\right],$$

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

$$p_s = \begin{cases} 2 \text{ dla } s = 0, \\ \\ 1 \text{ dla } s \neq 0. \end{cases}$$

Wprowadźmy dodatkowe $N/2$ na $M/2$-punktowe przekształcenia DST2D-II, określone na pewnych kombinacjach elementów $x(n,m)$ z wejściowej macierzy danych, postaci

$$X_0^{S2DII}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (x(2n,2m) + x(2n+1,2m) + x(2n,2m+1) +$$

$$+x(2n+1,2m+1)) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+1)(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., N/2-1$,

$$X_1^{S2DII}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (-1)^n (x(2n,2m) - x(2n+1,2m) + x(2n,2m+1) +$$

$$-x(2n+1,2m+1))) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+1)(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., N/2-1$,

$$X_2^{S2DII}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (-1)^m (x(2n,2m) + x(2n+1,2m) - x(2n,2m+1) -$$

$$-x(2n+1,2m+1)) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+1)(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., N/2-1$,

$$X_3^{S2DII}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (-1)^{m+n} (x(2n,2m) - x(2n+1,2m) - x(2n,2m+1) +$$

$$+x(2n+1,2m+1)) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+1)(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$. Stosując odpowiednie przekształcenia, które przeprowadza się w sposób zbliżony do pokazanego dla przypadku przekształcenia kosinusowego drugiego rodzaju, w rezultacie otrzymuje się wzór rozkładu szybkiego dwuetapowego algorytmu (FST2D-II) dla dyskretnego dwuwymiarowego przekształcenia sinusowego drugiego rodzaju. W celu uproszczenia zapisu wprowadźmy poniżej zdefiniowane symbole

$$C_N^{(k+1)} = cos\left(\frac{\pi}{2N}(k+1)\right),\ S_N^{(k+1)} = sin\left(\frac{\pi}{2N}(k+1)\right),$$

$$C_M^{(l+1)} = cos\left(\frac{\pi}{2M}(l+1)\right),\ S_M^{(l+1)} = sin\left(\frac{\pi}{2M}(l+1)\right).$$

Wówczas wzór rozkładu dwuetapowego algorytmu FST2D-II przyjmie postać

$$X_{N\cdot M}^{S2DII}(k,l) = X_0^{S2DII}(k,l)C_N^{(k+1)}C_M^{(l+1)} - X_1^{S2DII}(\frac{N}{2}-k-2,l)S_N^{(k+1)}C_M^{(l+1)} - \tag{4.33}$$
$$-X_2^{S2DII}(k,\frac{M}{2}-l-2)C_N^{(k+1)}S_M^{(l+1)} + X_3^{S2DII}(\frac{N}{2}-k-2,\frac{M}{2}-l-2)S_N^{(k+1)}S_M^{(l+1)}$$

dla $k = 0, 1, ..., N/2 - 2$ oraz $l = 0, 1, ..., M/2 - 2$. Poniżej zamieszczono dalszą postać wzoru rozkładu dla algorytmu FST2D-II, która rozpisana została z uwzględnieniem symetrii poszczególnych przekształceń składowych. Postać ta ułatwia implementację rozważanego szybkiego algorytmu obliczania przekształcenia DST2D-II.

$$\begin{aligned} X_{N\cdot M}^{S2DII}(N-k-2,l) &= X_0^{S2DII}(k,l)S_N^{(k+1)}C_M^{(l+1)} + \\ &+ X_1^{S2DII}(\tfrac{N}{2}-k-2,l)C_N^{(k+1)}C_M^{(l+1)} - X_2^{S2DII}(k,\tfrac{M}{2}-l-2)S_N^{(k+1)}S_M^{(l+1)} - \\ &- X_3^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2)C_N^{(k+1)}S_M^{(l+1)} \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 2$ i $l = 0, 1, ..., M/2 - 2$,

$$\begin{aligned} X_{N\cdot M}^{S2DII}(k,M-l-2) &= X_0^{S2DII}(k,l)C_N^{(k+1)}S_M^{(l+1)} - \\ &- X_1^{S2DII}(\tfrac{N}{2}-k-2,l)S_N^{(k+1)}S_M^{(l+1)} + X_2^{S2DII}(k,\tfrac{M}{2}-l-2)C_N^{(k+1)}C_M^{(l+1)} - \\ &- X_3^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2)S_N^{(k+1)}C_M^{(l+1)} \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 2$ i $l = 0, 1, ..., M/2 - 2$,

$$\begin{aligned} X_{N\cdot M}^{S2DII}(N-k-2,M-l-2) &= X_0^{S2DII}(k,l)S_N^{(k+1)}S_M^{(l+1)} + \\ &+ X_1^{S2DII}(\tfrac{N}{2}-k-2,l)C_N^{(k+1)}S_M^{(l+1)} + X_2^{S2DII}(k,\tfrac{M}{2}-l-2)S_N^{(k+1)}C_M^{(l+1)} + \\ &+ X_3^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2)C_N^{(k+1)}C_M^{(l+1)} \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 2$ i $l = 0, 1, ..., M/2 - 2$,

$$\begin{aligned} X_{N\cdot M}^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1) &= \tfrac{1}{2}X_0^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1), \\ X_{N\cdot M}^{S2DII}(N-1,\tfrac{M}{2}-1) &= \tfrac{\sqrt{2}}{2}X_1^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1), \\ X_{N\cdot M}^{S2DII}(\tfrac{N}{2}-1,M-1) &= \tfrac{\sqrt{2}}{2}X_2^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1), \\ X_{N\cdot M}^{SC2D(II)}(N-1,M-1) &= X_3^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1) \end{aligned}$$

oraz

$$\begin{aligned}
X_{N\cdot M}^{S2DII}(k,\tfrac{M}{2}-1) &= \tfrac{\sqrt{2}}{2}\Big(X_0^{S2DII}(k,\tfrac{M}{2}-1)C_N^{(k+1)}- \\
&- X_1^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)S_N^{(k+1)}\Big), \\
X_{N\cdot M}^{S2DII}(N-k-2,\tfrac{M}{2}-1) &= \tfrac{\sqrt{2}}{2}\Big(X_0^{S2DII}(k,\tfrac{M}{2}-1)S_N^{(k+1)}+ \\
&+ X_1^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)C_N^{(k+1)}\Big), \\
X_{N\cdot M}^{S2DII}(k,M-1) &= X_2^{S2DII}(k,\tfrac{M}{2}-1)C_N^{(k+1)}+ \\
&+ X_3^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2)S_N^{(k+1)}, \\
X_{N\cdot M}^{S2DII}(N-k-2,M-1) &= X_2^{S2DII}(k,\tfrac{M}{2}-1)S_N^{(k+1)}+ \\
&+ X_3^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)C_N^{(k+1)}
\end{aligned}$$

dla $k = 0, 1, ..., N/2 - 2$ oraz

$$\begin{aligned}
X_{N\cdot M}^{S2DII}(\tfrac{N}{2}-1,l) &= \tfrac{\sqrt{2}}{2}\Big(X_0^{S2DII}(\tfrac{N}{2}-1,l)C_M^{(l+1)}- \\
&- X_2^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)S_M^{(l+1)}\Big), \\
X_{N\cdot M}^{S2DII}(\tfrac{N}{2}-1,M-l-2) &= \tfrac{\sqrt{2}}{2}\Big(X_0^{SC2D(II)}(\tfrac{N}{2}-1,l)S_M^{(l+1)}+ \\
&+ X_2^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)C_M^{(l+1)}\Big), \\
X_{N\cdot M}^{S2DII}(N-1,l) &= X_1^{S2DII}(\tfrac{N}{2},l)C_M^{(l+1)}- \\
&- X_3^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)S_M^{(l+1)}, \\
X_{N\cdot M}^{S2DII}(N-1,M-l-2) &= X_1^{S2DII}(\tfrac{N}{2}-1,l)S_M^{(l+1)}+ \\
&+ X_3^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)C_M^{(l+1)}
\end{aligned}$$

dla $l = 0, 1, ..., M/2 - 2$.

Złożoność obliczeniowa szybkiego algorytmu dwuetapowego, który skonstruowano według wzoru rozkładu (4.33), jest rzędu $\mathcal{O}(N^2 log_2 N)$. Na wyznaczenie dokładnej liczby operacji rzeczywistych dodawań $Do_{N\cdot M}^{S2DII}$ oraz rzeczywistych mnożeń $Mn_{N\cdot M}^{S2DII}$, w zależności od rozmiaru N na M przekształcenia, pozwalają poniższe wzory

$$Do_{N\cdot M}^{S2DII} = \begin{cases} 6MN\left(\log_2 M + \frac{1}{3}\log_2 K - 1\right) - \frac{2}{3}M^2 + 4(M+N) - \frac{4}{3} & \text{dla } N \geqslant M, \\ 6MN\left(\log_2 N - \frac{1}{3}\log_2 K - 1\right) - \frac{2}{3}N^2 + 4(M+N) - \frac{4}{3} & \text{dla } N < M \end{cases}$$

oraz

$$Mn_{N\cdot M}^{S2DII} = \begin{cases} 8MN\left(\log_2 M + \frac{1}{4}\log_2 K - \frac{7}{4}\right) - 3M^2 + 11(M+N) - 5 & \text{dla } N \geqslant M, \\ 8MN\left(\log_2 N - \frac{1}{4}\log_2 K - \frac{7}{4}\right) - 3N^2 + 11(M+N) - 5 & \text{dla } N < M. \end{cases}$$

Występujący w powyższych zależnościach symbol K, obliczamy jako stosunek wartości N do M, tzn. jako $K = N/M$.

Jako ostatni rozpatrzony zostanie przypadek dyskretnego dwuwymiarowego przekształcenia sinusowego rodzaju czwartego.

Dwuwymiarowe dyskretne przekształcenie sinusowe czwartego rodzaju

Dyskretne dwuwymiarowe przekształcenie sinusowe czwartego (DST2D-IV) rodzaju, określone na macierzy elementów wejściowych $x(n,m)$ o wymiarze N wierszy na M kolumn, definiuje się zgodnie z zależnością

$$X_{N\cdot M}^{S2D(IV)}(k,l) = DST2DIV_{N\cdot M}\{x(n,m)\} \triangleq$$

$$\triangleq \frac{1}{NM}\left[\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x(n,m) sin\left(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{M}(l+\frac{1}{2})(m+\frac{1}{2})\right)\right]$$

dla $k = 0,1,...,N-1$ i $l = 0,1,...,M-1$. Po pominięciu czynnika normalizującego postaci $\frac{1}{NM}$, oraz po wprowadzeniu następujących przekształceń pomocniczych

$$X_0^{S2DIV}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (x(2n,2m) + x(2n+1,2m) + x(2n,2m+1) +$$

$$+x(2n+1,2m+1))sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0,1,...,N/2-1$ i $l = 0,1,...,N/2-1$,

$$X_1^{S2DIV}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (-1)^n(x(2n,2m) - x(2n+1,2m) + x(2n,2m+1) +$$

$$-x(2n+1,2m+1)))sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0,1,...,N/2-1$ i $l = 0,1,...,N/2-1$,

$$X_2^{S2DIV}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (-1)^m(x(2n,2m) + x(2n+1,2m) - x(2n,2m+1) -$$

$$-x(2n+1,2m+1))sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0,1,...,N/2-1$ i $l = 0,1,...,N/2-1$,

$$X_3^{S2DIV}(k,l) = \sum_{n=0}^{N/2-1}\sum_{m=0}^{M/2-1} (-1)^{m+n}(x(2n,2m) - x(2n+1,2m) - x(2n,2m+1) +$$

$$+x(2n+1,2m+1))sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., N/2 - 1$, a także dodatkowych symboli postaci

$$C_N^{(k+\frac{1}{2})} = cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right),\ S_N^{(k+\frac{1}{2})} = sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right),$$

$$C_M^{(l+\frac{1}{2})} = cos\left(\frac{\pi}{2M}(l+\frac{1}{2})\right),\ S_M^{(l+\frac{1}{2})} = sin\left(\frac{\pi}{2M}(l+\frac{1}{2})\right),$$

to wzór rozkładu szybkiego algorytmu (FSTIV-IV) dwuetapowego dla dyskretnego dwuwymiarowego przekształcenia sinusowego czwartego rodzaju można zapisać zgodnie z poniższą zależnością, jako

$$\begin{aligned} X_{N\cdot M}^{S2DIV}(k,l) &= X_0^{S2D(IV)}(k,l)C_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} - \\ &-X_1^{S2DIV}(\frac{N}{2}-k-1,l)S_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} - X_2^{S2DIV}(k,\frac{M}{2}-l-1)C_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} + \\ &+X_3^{S2DIV}(\frac{N}{2}-k-1,\frac{M}{2}-l-1)S_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} \end{aligned} \tag{4.34}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$. Dokładną postać tego wzoru rozpisaną według symetrii przekształceń składowych podano poniżej drobnym drukiem.

$$\begin{aligned} X_{N\cdot M}^{S2DIV}(N-k-1,l) &= X_0^{S2DIV}(k,l)S_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} + \\ &+ X_1^{S2DIV}(\tfrac{N}{2}-k-1,l)C_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} - X_2^{S2DIV}(k,\tfrac{M}{2}-l-1)S_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} - \\ &- X_3^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)C_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$,

$$\begin{aligned} X_{N\cdot M}^{S2DIV}(k,M-l-1) &= X_0^{S2DIV}(k,l)C_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} - \\ &- X_1^{S2DIV}(\tfrac{N}{2}-k-1,l)S_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} + X_2^{S2DIV}(k,\tfrac{M}{2}-l-1)C_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} - \\ &- X_3^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)S_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$ oraz

$$\begin{aligned} X_{N\cdot M}^{S2DIV}(N-k-1,M-l-1) &= X_0^{S2DIV}(k,l)S_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} + \\ &+ X_1^{S2DIV}(\tfrac{N}{2}-k-1,l)C_N^{(k+\frac{1}{2})}S_M^{(l+\frac{1}{2})} + X_2^{S2DIV}(k,\tfrac{M}{2}-l-1)S_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} + \\ &+ X_3^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)C_N^{(k+\frac{1}{2})}C_M^{(l+\frac{1}{2})} \end{aligned}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$.

Obliczanie N na M-punktowego przekształcenia DST2D-IV, przy użyciu algorytmu opisanego zależnością (4.34), odbywa się na podstawie czterech przekształceń $N/2$ na $M/2$-punktowych tego samego rodzaju. Stąd szybki algorytm dwuetapowy FST2D-IV, podobnie jak dotychczas opisane szybkie algorytmy dwuetapowe, charakteryzuje złożoność obliczeniowa rzędu $\mathcal{O}(N^2 log_2 N)$. Tutaj dokładną liczbę operacji rzeczywistych

dodawań $Do_{N\cdot M}^{S2DIV}$ oraz mnożeń $Mn_{N\cdot M}^{S2DIV}$ definiują następujące zależności

$$Do_{N\cdot M}^{S2DIV} = \begin{cases} 6MN\left(\log_2 M + \frac{1}{3}\log_2 K\right) & \text{dla } N \geqslant M, \\ 6MN\left(\log_2 N - \frac{1}{3}\log_2 K\right) & \text{dla } N < M \end{cases}$$

oraz

$$Mn_{N\cdot M}^{S2DIV} = \begin{cases} 8MN\left(\log_2 M + \frac{1}{4}\log_2 K + \frac{1}{8}\right) & \text{dla } N \geqslant M, \\ 8MN\left(\log_2 N - \frac{1}{4}\log_2 K + \frac{1}{8}\right) & \text{dla } N < M, \end{cases}$$

gdzie $K = N/M$.

Tym samym w niniejszej sekcji przedstawiono wzory rozkładów szybkich algorytmów dwuetapowych dla rozważanych przekształceń kosinusowych i sinusowych. Algorytmy te nie pozwalają jednak na konstrukcję szybkich algorytmów adaptacyjnego obliczania kosinusowego i sinusowego przekształcenia Fouriera, przez wzgląd na to, iż w każdym kroku działania szybkiego algorytmu dwuetapowego, przekształcenie N na M-punktowe obliczane jest na bazie czterech przekształceń, które nie operują bezpośrednio na elementach $x(n,m)$ macierzy wejściowej, lecz na pewnych ich kombinacjach.

W kolejnej sekcji przedstawiono schemat budowy szybkich algorytmów z przerzedzeniem w czasie na bazie algorytmów dwuetapowych, który zakłada zastosowanie innego niż $bit-reverse$ przemieszania elementów $x(n,m)$ wejściowej macierzy danych. Przedstawione algorytmy z przerzedzeniem w czasie spełniają warunki wymagane do budowy szybkich algorytmów adaptacyjnych dla całkowych przekształceń Fouriera.

4.2.3. Szybkie algorytmy z przerzedzeniem w czasie dla dyskretnych dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju

Przedstawione w poprzedniej sekcji algorytmy dwuetapowe operują na pewnych kombinacjach elementów $x(n,m)$ wejściowej macierzy danych i tym samym nie pozwalają na budowę szybkich algorytmów adaptacyjnego obliczania przekształceń całkowych. Podobnie jak w przypadku przekształceń jednowymiarowych, także i tutaj poprzez wprowadzenie specjalnego przemieszania elementów $x(n,m)$ wejściowej macierzy danych dla $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$, możliwe jest wyprowadzenie wzorów rozkładu szybkich algorytmów z przerzedzeniem w czasie. Za punkt wyjścia przyjęto przedstawione w poprzedniej sekcji szybkie algorytmy dwuetapowe. W tym celu wprowadźmy $N/2$ na $M/2$-elementowe ciągi $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$ złożone

z elementów $x(n,m)$ i definiowane według następujących reguł

$$\begin{aligned}
a(n,m) \triangleq\ & x(2n,2m)+x(2n+1,2m)+\\
+\ & (-1)^n\left(x(2n,2m)-x(2n+1,2m)\right)+\\
+\ & (-1)^m\left(x(2n,2m)+x(2n+1,2m)+\right.\\
+\ & \left.(-1)^n\left(x(2n,2m)-x(2n+1,2m)\right)\right)+\\
+\ & x(2n,2m+1)+x(2n+1,2m+1)+\\
+\ & (-1)^n\left(x(2n,2m+1)-x(2n+1,2m+1)\right)-\\
-\ & (-1)^m\left(x(2n,2m+1)+x(2n+1,2m+1)+\right.\\
+\ & \left.(-1)^n\left(x(2n,2m+1)-x(2n+1,2m+1)\right)\right)
\end{aligned} \tag{4.35}$$

dla $n=0,1,...,N/2-1$ i $m=0,1,...,M/2-1$,

$$\begin{aligned}
b(n,m) \triangleq\ & x(2n,2m)+x(2n+1,2m)-\\
-\ & (-1)^n\left(x(2n,2m)-x(2n+1,2m)\right)+\\
+\ & (-1)^m\left(x(2n,2m)+x(2n+1,2m)-\right.\\
-\ & \left.(-1)^n\left(x(2n,2m)-x(2n+1,2m)\right)\right)+\\
+\ & x(2n,2m+1)+x(2n+1,2m+1)-\\
-\ & (-1)^n\left(x(2n,2m+1)-x(2n+1,2m+1)\right)-\\
-\ & (-1)^m\left(x(2n,2m+1)+x(2n+1,2m+1)-\right.\\
-\ & \left.(-1)^n\left(x(2n,2m+1)-x(2n+1,2m+1)\right)\right)
\end{aligned}$$

dla $n=0,1,...,N/2-1$ i $m=0,1,...,M/2-1$,

$$\begin{aligned}
c(n,m) \triangleq\ & x(2n,2m)+x(2n+1,2m)+\\
+\ & (-1)^n\left(x(2n,2m)-x(2n+1,2m)\right)-
\end{aligned}$$

$$
\begin{aligned}
&- \ (-1)^m \left(x(2n, 2m) + x(2n+1, 2m) + \right. \\
&+ \ (-1)^n \left(x(2n, 2m) - x(2n+1, 2m)\right)\Big) + \\
&+ \ x(2n, 2m+1) + x(2n+1, 2m+1) + \\
&+ \ (-1)^n \left(x(2n, 2m+1) - x(2n+1, 2m+1)\right) + \\
&+ \ (-1)^m \left(x(2n, 2m+1) + x(2n+1, 2m+1) + \right. \\
&+ \ (-1)^n \left(x(2n, 2m+1) - x(2n+1, 2m+1)\right)\Big)
\end{aligned}
$$

dla $n = 0, 1, ..., N/2 - 1$ i $m = 0, 1, ..., M/2 - 1$ oraz

$$
\begin{aligned}
d(n,m) \ \triangleq \ & x(2n, 2m) + x(2n+1, 2m) - \\
&- \ (-1)^n \left(x(2n, 2m) - x(2n+1, 2m)\right) - \\
&- \ (-1)^m \left(x(2n, 2m) + x(2n+1, 2m) - \right. \\
&- \ (-1)^n \left(x(2n, 2m) - x(2n+1, 2m)\right)\Big) + \\
&+ \ x(2n, 2m+1) + x(2n+1, 2m+1) - \\
&- \ (-1)^n \left(x(2n, 2m+1) - x(2n+1, 2m+1)\right) + \\
&+ \ (-1)^m \left(x(2n, 2m+1) + x(2n+1, 2m+1) - \right. \\
&- \ (-1)^n \left(x(2n, 2m+1) - x(2n+1, 2m+1)\right)\Big)
\end{aligned}
$$

dla $n = 0, 1, ..., N/2 - 1$ i $m = 0, 1, ..., M/2 - 1$. Poprzez odpowiednie pogrupowanie składników ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$ względem indeksów n i m, powyższe wyrażenia można zapisać w alternatywnej postaci jako

$$
a(n,m) = \begin{cases}
x(2n, 2m) & \text{dla } n \text{ i } m \text{ parzystych ,} \\
x(2n+1, 2m) & \text{dla } n \text{ nieparzystych i } m \text{ parzystych,} \\
x(2n, 2m+1) & \text{dla } n \text{ parzystych i } m \text{ nieparzystych ,} \\
x(2n+1, 2m+1) & \text{dla } n \text{ i } m \text{ nieparzystych ,}
\end{cases} \tag{4.36}
$$

gdzie $n = 0, 1, ..., N/2 - 1$ i $m = 0, 1, ..., M/2 - 1$,

$$b(n,m) = \begin{cases} x(2n+1, 2m) & \text{dla } n \text{ i } m \text{ parzystych ,} \\ x(2n, 2m) & \text{dla } n \text{ nieparzystych i } m \text{ parzystych,} \\ x(2n+1, 2m+1) & \text{dla } n \text{ parzystych i } m \text{ nieparzystych ,} \\ x(2n, 2m+1) & \text{dla } n \text{ i } m \text{ nieparzystych ,} \end{cases}$$

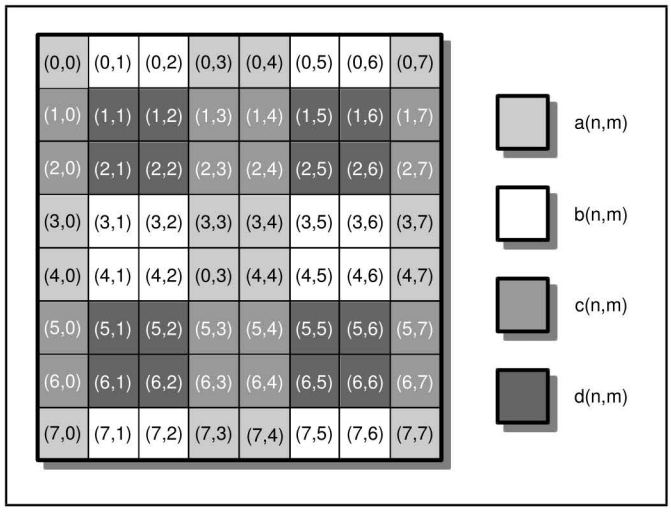
gdzie $n = 0, 1, ..., N/2 - 1$ i $m = 0, 1, ..., M/2 - 1$,

$$c(n,m) = \begin{cases} x(2n, 2m+1) & \text{dla } n \text{ i } m \text{ parzystych ,} \\ x(2n+1, 2m+1) & \text{dla } n \text{ nieparzystych i } m \text{ parzystych,} \\ x(2n, 2m) & \text{dla } n \text{ parzystych i } m \text{ nieparzystych ,} \\ x(2n+1, 2m) & \text{dla } n \text{ i } m \text{ nieparzystych ,} \end{cases}$$

gdzie $n = 0, 1, ..., N/2 - 1$ i $m = 0, 1, ..., M/2 - 1$ oraz

$$d(n,m) = \begin{cases} x(2n+1, 2m+1) & \text{dla } n \text{ i } m \text{ parzystych ,} \\ x(2n, 2m+1) & \text{dla } n \text{ nieparzystych i } m \text{ parzystych,} \\ x(2n+1, 2m) & \text{dla } n \text{ parzystych i } m \text{ nieparzystych ,} \\ x(2n, 2m) & \text{dla } n \text{ i } m \text{ nieparzystych ,} \end{cases}$$

gdzie $n = 0, 1, ..., N/2-1$ i $m = 0, 1, ..., M/2-1$. Z powyższych zależności wynika, iż elementami nowo wprowadzonych ciągów, definiowanych jako (4.35), są odpowiednio przemieszane elementy $x(n, m)$ macierzy wejściowej. Ponadto łatwo wykazać, iż ciągi te nie posiadają elementów wspólnych, tj. elementów o tych samych indeksach n oraz m, i w ten sposób wykorzystują cały dostępny zbiór elementów $x(n, m)$ dla $n=0, 1, ..., N-1$ i $m = 0, 1, ..., M - 1$. Rysunek 4.7 prezentuje sposób doboru elementów $x(n, m)$



Rysunek 4.7: Sposób doboru elementów $x(n,m)$ dla ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$ w przypadku, gdy $N = M = 8$

dla ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$ dla przypadku $N = 8$ na $M = 8$ elementowej macierzy wejściowej.

Proces syntezy szybkich algorytmów z przerzedzeniem w czasie dla dyskretnych dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju opiera się na następujących własnościach ciągów definiowanych jako (4.35)

$$\begin{aligned} &a(n,m) + b(n,m) + c(n,m) + d(n,m) = \\ &= x(2n,2m) + x(2n+1,2m) + x(2n,2m+1) + x(2n+1,2m+1) \end{aligned} \tag{4.37}$$

dla $n = 0,1,...,N/2-1$ i $m = 0,1,...,M/2-1$,

$$a(n,m) - b(n,m) + c(n,m) - d(n,m) =$$

$$= (-1)^n \left(x(2n,2m) - x(2n+1,2m) + x(2n,2m+1) - x(2n+1,2m+1)\right)$$

dla $n = 0,1,...,N/2-1$ i $m = 0,1,...,M/2-1$,

$$a(n,m) + b(n,m) - c(n,m) - d(n,m) =$$

$$= (-1)^m \left(x(2n,2m) + x(2n+1,2m) - x(2n,2m+1) - x(2n+1,2m+1)\right)$$

dla $n = 0,1,...,N/2-1$ i $m = 0,1,...,M/2-1$,

$$a(n,m) - b(n,m) - c(n,m) + d(n,m) =$$

$$= (-1)^n (-1)^m \left(x(2n,2m) - x(2n+1,2m) - x(2n,2m+1) + x(2n+1,2m+1)\right)$$

dla $n = 0, 1, ..., N/2 - 1$ i $m = 0, 1, ..., M/2 - 1$. Własności (4.37) wskazują na to, iż sumy i różnice elementów ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ i $d(n,m)$ o tych samych indeksach, stanowią kombinacje elementów macierzy $x(n,m)$, na których określone były przekształcenia X_0, X_1, X_2 i X_3 we wzorach rozkładu szybkich algorytmów dwuetapowych. W przeciwieństwie do tych kombinacji przekształcenia operujące na sumach i różnicach elementów $a(n,m)$, $b(n,m)$, $c(n,m)$ i $d(n,m)$, zgodnie z własnością liniowości rozpatrywanych przekształceń trygonometrycznych (patrz [9, 14, 42, 94, 136]), można rozpisać jako sumy i różnice przekształceń operujących bezpośrednio na elementach $a(n,m)$, $b(n,m)$, $c(n,m)$ i $d(n,m)$, a co za tym idzie, operujących bezpośrednio na elementach $x(n,m)$ macierzy danych wejściowych.

Dwuwymiarowe dyskretne przekształcenie kosinusowe drugiego rodzaju

Wprowadźmy pomocnicze $N/2$ na $M/2$-punktowe przekształcenia kosinusowe drugiego rodzaju, które operują odpowiednio na elementach $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$

$$X_4^{C2DII}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} a(n,m) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)} k(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)} l(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$,

$$X_5^{C2DII}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} b(n,m) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)} k(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)} l(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$,

$$X_6^{C2DII}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} c(n,m) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)} k(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)} l(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$,

$$X_7^{C2DII}(k,l) = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} d(n,m) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)} k(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)} l(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$. Wówczas na mocy Własności (4.37) ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ i $d(n,m)$, oraz liniowości rozpatrywanych przekształceń trygonometrycznych (patrz [9, 14, 42, 94, 136]), wzór rozkładu szybkiego algorytmu dwuetapowego dla dyskretnego dwuwymiarowego przekształcenia kosinusowego drugiego rodzaju (4.31) można zapisać w postaci

$$\begin{aligned} X_{N\cdot M}^{C2DII}(k,l) &= \Big(X_4^{C2DII}(k,l) + X_5^{C2DII}(k,l) + \\ &+ X_6^{C2DII}(k,l) + X_7^{C2DII}(k,l)\Big) C_N^k C_M^l + \\ &+ \Big(X_4^{C2DII}(\frac{N}{2}-k,l) - X_5^{C2DII}(\frac{N}{2}-k,l) + \\ &+ X_6^{C2DII}(\frac{N}{2}-k,l) - X_7^{C2DII}(\frac{N}{2}-k,l)\Big) S_N^k C_M^l + \end{aligned} \tag{4.38}$$

$$
\begin{aligned}
&+ \Big(X_4^{C2DII}(k, \frac{M}{2} - l) + X_5^{C2DII}(k, \frac{M}{2} - l) - \\
&- X_6^{C2DII}(k, \frac{M}{2} - l) - X_7^{C2DII}(k, \frac{M}{2} - l) \Big) C_N^k S_M^l + \\
&+ \Big(X_4^{C2DII}(\frac{N}{2} - k, \frac{M}{2} - l) - X_5^{C2DII}(\frac{N}{2} - k, \frac{M}{2} - l) - \\
&- X_6^{C2DII}(\frac{N}{2} - k, \frac{M}{2} - l) + X_7^{C2DII}(\frac{N}{2} - k, \frac{M}{2} - l) \Big) S_N^k S_M^l
\end{aligned}
$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$, gdzie

$$
C_N^k = cos\left(\frac{\pi}{2N}k\right), \; S_N^k = sin\left(\frac{\pi}{2N}k\right),
$$

$$
C_M^l = cos\left(\frac{\pi}{2M}l\right), \; S_M^l = sin\left(\frac{\pi}{2M}l\right).
$$

Z analizy powyższej zależności wynika, iż do obliczenia N na M-punktowego przekształcenia DCT2D-II wymagane są tutaj cztery przekształcenia $N/2$ na $M/2$-punktowe tego samego typu, które operują na ciągach $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$, tzn. bezpośrednio na elementach $x(n,m)$ wejściowej macierzy danych (patrz (4.36)). Stąd można jednoznacznie stwierdzić, iż otrzymany poprzez wprowadzenie ciągów (4.35) szybki algorytm, jest algorytmem z przerzedzeniem w czasie, a co za tym idzie posiada on strukturę odpowiednią do budowy szybkiego algorytmu adaptacyjnego obliczania całkowego kosinusowego przekształcenia Fouriera.

Zależność (4.38) prezentuje wzór rozkładu szybkiego algorytmu (FCT2D-II) z przerzedzeniem w czasie, który pozwala na jego implementację programową. Jednakże poniżej drobnym drukiem zamieszczono postać tego wzoru rozpisaną względem własności symetrii przekształceń składowych. Postać ta jest znacznie wygodniejsza i pozwala na łatwiejszą implementację szybkiego algorytmu.

$$
\begin{aligned}
X_{N\cdot M}^{C2DII}(N-k,l) &= -\big(X_4^{C2DII}(k,l) + X_5^{C2DII}(k,l) + \\
&+ X_6^{C2DII}(k,l) + X_7^{C2DII}(k,l) \big) S_N^k C_M^l + \\
&+ \big(X_4^{C2DII}(\tfrac{N}{2}-k,l) - X_5^{C2DII}(\tfrac{N}{2}-k,l) + \\
&+ X_6^{C2DII}(\tfrac{N}{2}-k,l) - X_7^{C2DII}(\tfrac{N}{2}-k,l) \big) C_N^k C_M^l - \\
&- \big(X_4^{C2DII}(k,\tfrac{M}{2}-l) + X_5^{C2DII}(k,\tfrac{M}{2}-l) - \\
&- X_6^{C2DII}(k,\tfrac{M}{2}-l) - X_7^{C2DII}(k,\tfrac{M}{2}-l) \big) S_N^k S_M^l + \\
&+ \big(X_4^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) - X_5^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) + \\
&+ X_6^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) + X_7^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) \big) C_N^k S_M^l
\end{aligned}
$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$,

$$
\begin{aligned}
X_{N\cdot M}^{C2DII}(k,M-l) &= -\big(X_4^{C2DII}(k,l) + X_5^{C2DII}(k,l) + \\
&+ X_6^{C2DII}(k,l) + X_7^{C2DII}(k,l) \big) C_N^k S_M^l -
\end{aligned}
$$

$$
\begin{aligned}
- \quad & \big(X_4^{C2DII}(\tfrac{N}{2}-k,l) - X_5^{C2DII}(\tfrac{N}{2}-k,l) + \\
+ \quad & X_6^{C2DII}(\tfrac{N}{2}-k,l) - X_7^{C2DII}(\tfrac{N}{2}-k,l)\big) S_N^k S_M^l + \\
+ \quad & \big(X_4^{C2DII}(k,\tfrac{M}{2}-l) + X_5^{C2DII}(k,\tfrac{M}{2}-l) - \\
- \quad & X_6^{C2DII}(k,\tfrac{M}{2}-l) - X_7^{C2DII}(k,\tfrac{M}{2}-l)\big) C_N^k C_M^l + \\
+ \quad & \big(X_4^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) - X_5^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) + \\
+ \quad & X_6^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) + X_7^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l)\big) S_N^k C_M^l
\end{aligned}
$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$,

$$
\begin{aligned}
X_{N\cdot M}^{C2DII}(N-k,M-l) = \quad & \big(X_4^{C2DII}(k,l) + X_5^{C2DII}(k,l) + \\
+ \quad & X_6^{C2DII}(k,l) + X_7^{C2DII}(k,l)\big) S_N^k S_M^l - \\
- \quad & \big(X_4^{C2DII}(\tfrac{N}{2}-k,l) - X_5^{C2DII}(\tfrac{N}{2}-k,l) + \\
+ \quad & X_6^{C2DII}(\tfrac{N}{2}-k,l) - X_7^{C2DII}(\tfrac{N}{2}-k,l)\big) C_N^k S_M^l - \\
- \quad & \big(X_4^{C2DII}(k,\tfrac{M}{2}-l) + X_5^{C2DII}(k,\tfrac{M}{2}-l) - \\
- \quad & X_6^{C2DII}(k,\tfrac{M}{2}-l) - X_7^{C2DII}(k,\tfrac{M}{2}-l)\big) S_N^k C_M^l + \\
+ \quad & \big(X_4^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) - X_5^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) + \\
+ \quad & X_6^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l) + X_7^{C2DII}(\tfrac{N}{2}-k,\tfrac{M}{2}-l)\big) C_N^k C_M^l
\end{aligned}
$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$ oraz

$$
\begin{aligned}
X_{N\cdot M}^{C2DII}(0,0) &= \big(X_4^{C2DII}(0,0) + X_5^{C2DII}(0,0) + \\
&+ X_6^{C2DII}(0,0) + X_7^{C2DII}(0,0)\big), \\
X_{N\cdot M}^{C2DII}(\tfrac{N}{2},0) &= \tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(0,0) - X_5^{C2DII}(0,0) + \\
&+ X_6^{C2DII}(0,0) - X_7^{C2DII}(0,0)\big), \\
X_{N\cdot M}^{C2DII}(0,\tfrac{M}{2}) &= \tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(0,0) + X_5^{C2DII}(0,0) - \\
&- X_6^{C2DII}(0,0) - X_7^{C2DII}(0,0)\big), \\
X_{N\cdot M}^{C2DII}(\tfrac{N}{2},\tfrac{M}{2}) &= \tfrac{1}{2}\big(X_4^{C2DII}(0,0) - X_5^{C2DII}(0,0) - \\
&- X_6^{C2DII}(0,0) + X_7^{C2DII}(0,0)\big),
\end{aligned}
$$

oraz

$$
\begin{aligned}
X_{N\cdot M}^{C2DII}(k,0) &= \big(X_4^{C2DII}(k,0) + X_5^{C2DII}(k,0) + \\
&+ X_6^{C2DII}(k,0) + X_7^{C2DII}(k,0)\big) C_N^k + \\
&+ \big(X_4^{C2DII}(\tfrac{N}{2}-k,0) - X_5^{C2DII}(\tfrac{N}{2}-k,0) + \\
&+ X_6^{C2DII}(\tfrac{N}{2}-k,0) - X_7^{C2DII}(\tfrac{N}{2}-k,0)\big) S_N^k, \\
X_{N\cdot M}^{C2DII}(N-k,0) &= -\big(X_4^{C2DII}(k,0) + X_5^{C2DII}(k,0) + \\
&+ X_6^{C2DII}(k,0) + X_7^{C2DII}(k,0)\big) S_N^k +
\end{aligned}
$$

$$
\begin{aligned}
X_{N\cdot M}^{C2DII}(k,\tfrac{M}{2}) &= \tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(k,0)+X_5^{C2DII}(k,0)- \\
&+ \big(X_4^{C2DII}(\tfrac{N}{2}-k,0)-X_5^{C2DII}(\tfrac{N}{2}-k,0)+ \\
&+ X_6^{C2DII}(\tfrac{N}{2}-k,0)-X_7^{C2DII}(\tfrac{N}{2}-k,0)\big)C_N^k, \\
&- X_6^{C2DII}(k,0)-X_7^{C2DII}(k,0)\big)C_N^k+ \\
&+ \tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(\tfrac{N}{2}-k,0)-X_5^{C2DII}(\tfrac{N}{2}-k,0)- \\
&- X_6^{C2DII}(\tfrac{N}{2}-k,0)+X_7^{C2DII}(\tfrac{N}{2}-k,0)\big)S_N^k, \\
X_{N\cdot M}^{C2DII}(N-k,\tfrac{M}{2}) &= -\tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(k,0)+X_5^{C2DII}(k,0)- \\
&- X_6^{C2DII}(k,0)-X_7^{C2DII}(k,0)\big)S_N^k+ \\
&+ \tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(\tfrac{N}{2}-k,0)-X_5^{C2DII}(\tfrac{N}{2}-k,0)- \\
&- X_6^{C2DII}(\tfrac{N}{2}-k,0)+X_7^{C2DII}(\tfrac{N}{2}-k,0)\big)C_N^k
\end{aligned}
$$

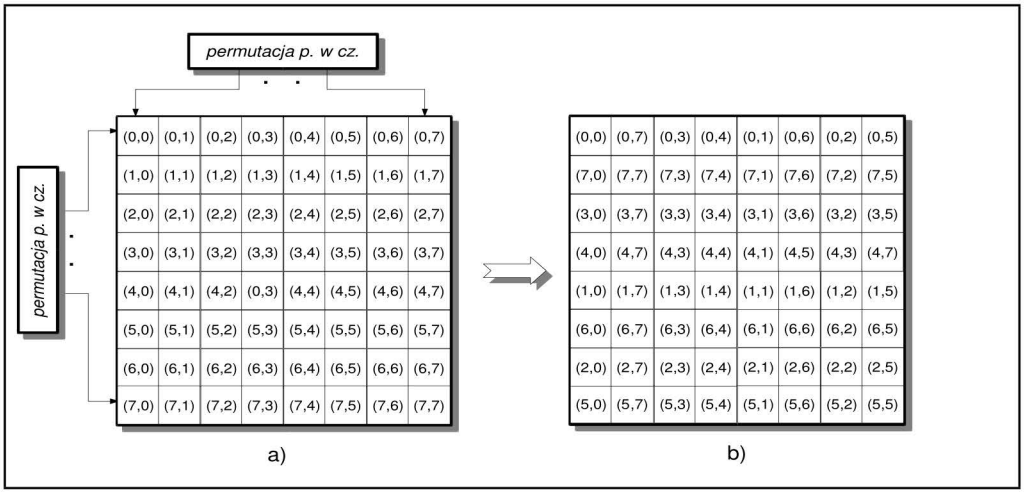
dla $k = 0, 1, ..., N/2-1$ oraz

$$
\begin{aligned}
X_{N\cdot M}^{C2DII}(0,l) &= \big(X_4^{C2DII}(0,l)+X_5^{C2DII}(0,l)+ \\
&+ X_6^{C2DII}(0,l)+X_7^{C2DII}(0,l)\big)C_M^l+ \\
&+ \big(X_4^{C2DII}(0,\tfrac{M}{2}-l)-X_5^{C2DII}(0,\tfrac{M}{2}-l)+ \\
&+ X_6^{C2DII}(0,\tfrac{M}{2}-l)-X_7^{C2DII}(0,\tfrac{M}{2}-l)\big)S_M^l, \\
X_{N\cdot M}^{C2DII}(0,M-l) &= -\big(X_4^{C2DII}(0,l)+X_5^{C2DII}(0,l)+ \\
&+ X_6^{C2DII}(0,l)+X_7^{C2DII}(0,l)\big)S_M^l+ \\
&+ \big(X_4^{C2DII}(0,\tfrac{M}{2}-l)-X_5^{C2DII}(0,\tfrac{M}{2}-l)+ \\
&+ X_6^{C2DII}(0,\tfrac{M}{2}-l)-X_7^{C2DII}(0,\tfrac{M}{2}-l)\big)C_M^l, \\
X_{N\cdot M}^{C2DII}(\tfrac{N}{2},l) &= \tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(0,l)-X_5^{C2DII}(0,l)+ \\
&+ X_6^{C2DII}(0,l)-X_7^{C2DII}(0,l)\big)C_M^l+ \\
&+ \tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(0,\tfrac{M}{2}-l)-X_5^{C2DII}(0,\tfrac{M}{2}-l)- \\
&- X_6^{C2DII}(0,\tfrac{M}{2}-l)+X_7^{C2DII}(0,\tfrac{M}{2}-l)\big)S_M^l, \\
X_{N\cdot M}^{C2DII}(\tfrac{N}{2},M-l) &= -\tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(0,l)-X_5^{C2DII}(0,l)+ \\
&+ X_6^{C2DII}(0,l)-X_7^{C2DII}(0,l)\big)S_M^l+ \\
&+ \tfrac{\sqrt{2}}{2}\big(X_4^{C2DII}(0,\tfrac{M}{2}-l)-X_5^{C2DII}(0,\tfrac{M}{2}-l)- \\
&- X_6^{C2DII}(0,\tfrac{M}{2}-l)+X_7^{C2DII}(0,\tfrac{M}{2}-l)\big)C_M^l
\end{aligned}
$$

dla $l = 0, 1, ..., M/2-1$.

Rekurencyjne użycie zależności (4.38) umożliwia szybkie obliczanie dwuwymiarowego przekształcenia DCT2D-II dla N i M będących potęgami dwóch. W związku z wykorzystaniem ciągów pomocniczych $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$, które zawierają elementy $x(n,m)$ o odpowiednio dobranych indeksach n i m, na wejście szybkiego algorytmu z przerzedzeniem w czasie podaje się macierz elementów $x(n,m)$ przemieszaną zgodnie z pewną specyficzną kolejnością, inną niż kolejność $bit-reverse$. Rysunek 4.8 przedstawia sposób przemieszania elementów dla przypadku $N = M = 8$. Natomiast w dodatku A zamieszczono procedurę realizującą wymagane przemieszanie elementów dla szybkich algorytmów z przerzedzeniem w czasie (patrz proc. A.10).

Ponieważ wzór rozkładu szybkiego algorytmu z przerzedzeniem w czasie (4.38) został wyprowadzony na bazie szybkiego algorytmu dwuetapowego (4.31), jedynie po-



Rysunek 4.8: Sposób przemieszania elementów macierzy danych (a) wejściowych dla algorytmu z przerzedzeniem w czasie dla przypadku $N = M = 8$ (b)

przez zastosowanie specjalnego przemieszania elementów wejściowych, to charakteryzuje go ta sama złożoność obliczeniowa co algorytm dwuetapowy. Zatem na wyznaczenie dokładnej liczby operacji rzeczywistych dodawań $Do_{N\cdot M}^{C2DII}$ i mnożeń $Mn_{N\cdot M}^{C2DII}$ dla algorytmu FCT2D-II z przerzedzeniem w czasie, pozwalają zależności podane w poprzedniej sekcji dla algorytmu dwuetapowego.

W dodatku A zamieszczono procedurę będącą implementacją szybkiego algorytmu (patrz wzór (4.38)) obliczania dwuwymiarowego przekształcenia kosinusowego drugiego rodzaju (patrz proc. A.11).

Dwuwymiarowe dyskretne przekształcenie kosinusowe czwartego rodzaju

W analogiczny sposób konstruuje się szybki algorytm z przerzedzeniem w czasie dla dwuwymiarowego dyskretnego N na M - punktowego przekształcenia kosinusowego czwartego rodzaju. W tym celu do przedstawionego we wcześniejszym paragrafie wzoru rozkładu szybkiego algorytmu dwuetapowego, wprowadzamy pomocnicze ciągi $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$ dla $n = 0,1,...,N/2-1$ i $m = 0,1,...,M-1$, definiowane jako (4.35), oraz posiadające własności postaci (4.37). Dla uproszczenia zapisu wprowadzamy nastepujące przekształcenia $N/2$ na $M/2$-punktowe, które operują na wspomnianych ciągach

$$X_4^{C2DIV}(k,l) =$$

$$= \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} a(n,m) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ oraz $l = 0, 1, ..., M/2 - 1$,

$$X_5^{C2DIV}(k,l) =$$

$$= \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} b(n,m) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ oraz $l = 0, 1, ..., M/2 - 1$,

$$X_6^{C2DIV}(k,l) =$$

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

dla $k = 0, 1, ..., N/2 - 1$ oraz $l = 0, 1, ..., M/2 - 1$ i

$$X_7^{C2DIV}(k,l) =$$

$$= \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} d(n,m) cos\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) cos\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ oraz $l = 0, 1, ..., M/2 - 1$. Wówczas wzór rozkładu szybkiego algorytmu z przerzedzeniem w czasie dla dwuwymiarowego dyskretnego przekształcenia kosinusowego czwartego rodzaju przyjmuje postać

$$\begin{aligned}
X_{N\cdot M}^{C2DIV}(k,l) &= (X_4^{C2DIV}(k,l) + X_5^{C2DIV}(k,l) + \\
&+X_6^{C2DIV}(k,l) + X_7^{C2DIV}(k,l))C_N^{k+\frac{1}{2}} C_M^{l+\frac{1}{2}} + \\
&+(X_4^{C2DIV}(\frac{N}{2}-k-1,l) - X_5^{C2DIV}(\frac{N}{2}-k-1,l) + \\
&+X_6^{C2DIV}(\frac{N}{2}-k-1,l) - X_7^{C2DIV}(\frac{N}{2}-k-1,l))S_N^{k+\frac{1}{2}} C_M^{l+\frac{1}{2}} + \\
&+(X_4^{C2DIV}(k,\frac{M}{2}-l-1) + X_5^{C2DIV}(k,\frac{M}{2}-l-1) - \\
&-X_6^{C2DIV}(k,\frac{M}{2}-l-1) - X_7^{C2DIV}(k,\frac{M}{2}-l-1))C_N^{k+\frac{1}{2}} S_M^{l+\frac{1}{2}} + \\
&+(X_4^{C2DIV}(\frac{N}{2}-k-1,\frac{N}{2}-l-1) - X_5^{C2DIV}(\frac{N}{2}-k-1,\frac{M}{2}-l-1) - \\
&-X_6^{C2DIV}(\frac{N}{2}-k-1,\frac{N}{2}-l-1) + X_7^{C2DIV}(\frac{N}{2}-k-1,\frac{M}{2}-l-1))S_N^{k+\frac{1}{2}} S_M^{l+\frac{1}{2}}
\end{aligned} \tag{4.39}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$, oraz

$$
\begin{aligned}
X_{N\cdot M}^{C2DIV}(N-k-1,l) &= -(X_4^{C2DIV}(k,l)+X_5^{C2DIV}(k,l)+\\
&+ X_6^{C2DIV}(k,l)+X_7^{C2DIV}(k,l))S_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}+\\
&+ (X_4^{C2DIV}(\tfrac{N}{2}-k-1,l)-X_5^{C2DIV}(\tfrac{N}{2}-k-1,l)+\\
&+ X_6^{C2DIV}(\tfrac{N}{2}-k-1,l)-X_7^{C2DIV}(\tfrac{N}{2}-k-1,l))C_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}-\\
&- (X_4^{C2DIV}(k,\tfrac{M}{2}-l-1)+X_5^{C2DIV}(k,\tfrac{M}{2}-l-1)-\\
&- X_6^{C2DIV}(k,\tfrac{M}{2}-l-1)-X_7^{C2DIV}(k,\tfrac{M}{2}-l-1))S_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}+\\
&+ (X_4^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)-X_5^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)-\\
&- X_6^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)+X_7^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1))C_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}
\end{aligned}
$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$,

$$
\begin{aligned}
X_{N\cdot M}^{C2DIV}(k,M-l-1) &= -(X_4^{C2DIV}(k,l)+X_5^{C2DIV}(k,l)+\\
&+ X_6^{C2DIV}(k,l)+X_7^{C2DIV}(k,l))C_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}-\\
&- (X_4^{C2DIV}(\tfrac{N}{2}-k-1,l)-X_5^{C2DIV}(\tfrac{N}{2}-k-1,l)+\\
&+ X_6^{C2DIV}(\tfrac{N}{2}-k-1,l)-X_7^{C2DIV}(\tfrac{N}{2}-k-1,l))S_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}+\\
&+ (X_4^{C2DIV}(k,\tfrac{M}{2}-l-1)+X_5^{C2DIV}(k,\tfrac{M}{2}-l-1)-\\
&- X_6^{C2DIV}(k,\tfrac{M}{2}-l-1)-X_7^{C2DIV}(k,\tfrac{M}{2}-l-1))C_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}+\\
&+ (X_4^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)-X_5^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)-\\
&- X_6^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)+X_7^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1))S_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}
\end{aligned}
$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$ oraz

$$
\begin{aligned}
X_{N\cdot M}^{C2DIV}(N-k-1,M-l-1) &= (X_4^{C2DIV}(k,l)+X_5^{C2DIV}(k,l)+\\
&+ X_6^{C2DIV}(k,l)+X_7^{C2DIV}(k,l))S_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}-\\
&- (X_4^{C2DIV}(\tfrac{N}{2}-k-1,l)-X_5^{C2DIV}(\tfrac{N}{2}-k-1,l)+\\
&+ X_6^{C2DIV}(\tfrac{N}{2}-k-1,l)-X_7^{C2DIV}(\tfrac{N}{2}-k-1,l))C_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}-\\
&- (X_4^{C2DIV}(k,\tfrac{M}{2}-l-1)+X_5^{C2DIV}(k,\tfrac{M}{2}-l-1)-\\
&- X_6^{C2DIV}(k,\tfrac{M}{2}-l-1)-X_7^{C2DIV}(k,\tfrac{M}{2}-l-1))S_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}+\\
&+ (X_4^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)-X_5^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)-\\
&- X_6^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)+X_7^{C2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1))C_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}
\end{aligned}
$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$,

gdzie

$$C_N^{k+\frac{1}{2}} = cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right),\ S_N^{k+\frac{1}{2}} = sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right),$$

$$C_M^{l+\frac{1}{2}} = cos\left(\frac{\pi}{2M}(l+\frac{1}{2})\right),\ S_M^{l+\frac{1}{2}} = sin\left(\frac{\pi}{2M}(l+\frac{1}{2})\right).$$

Z powyższych zależności wynika, iż według proponowanego szybkiego algorytmu z przerzedzeniem w czasie N na M punktowe przekształcenie DCT2D-IV obliczane

jest na podstawie czterech przekształceń $N/2$ na $M/2$ punktowych tego samego rodzaju, które operują bezpośrednio na elementach wejściowej macierzy danych $x(n,m)$. Rekurencyjne użycie wzoru (4.39) umożliwia budowę szybkiego algorytmu z przerzedzeniem w czasie dla przekształcenia kosinusowego czwartego rodzaju o wymiarze N na M, gdzie N i M są potęgami dwóch. Elementy wejściowej macierzy danych $x(n,m)$ muszą być wówczas przemieszane zgodnie z kolejnością wynikającą ze sposobu doboru próbek $x(n,m)$ dla ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$ (patrz rysunek 4.8). Wymagane przemieszanie realizuje procedura A.10, natomiast implementację szybkiego algorytmu zbudowanego zgodnie ze wzorem rozkładu (4.39) zawiera procedura A.12.

Ponieważ szybki algorytm z przerzedzeniem w czasie powstał na bazie szybkiego algorytmu dwuetapowego jedynie poprzez wprowadzenie innego niż $bit-reverse$ przemieszania elementów macierzy danych wejściowych, to charakteryzuje go ta sama złożoność obliczeniowa co wspomniany szybki algorytm dwuetapowy. Zatem do wyznaczenia dokładnej liczby rzeczywistych operacji dodawania $Do_{N\cdot M}^{C2DIV}$ i mnożenia $Mn_{N\cdot M}^{C2DIV}$ posłużyć mogą zależności podane w poprzedniej sekcji.

Dwuwymiarowe dyskretne przekształcenie sinusowe drugiego typu

Przez analogię do poprzednio prezentowanych przekształceń, również i w tym przypadku wprowadźmy $N/2$ na $M/2$ punktowe pomocnicze przekształcenia sinusowe drugiego typu, które operują odpowiednio na elementach ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ oraz $d(n,m)$

$$X_4^{S2DII}(k,l) =$$

$$= \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} a(n,m) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+1)(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$,

$$X_5^{S2DII}(k,l) =$$

$$= \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} b(n,m) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+1)(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$,

$$X_6^{S2DII}(k,l) =$$

$$= \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} c(n,m) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+1)(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$ i

$$X_7^{S2DII}(k,l) =$$

$$= \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} d(n,m) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+1)(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+1)(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2-1$ i $l = 0, 1, ..., M/2-1$. Przy pomocy wprowadzonych przekształceń pomocniczych, oraz na mocy własności (4.37) ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ i $d(n,m)$, definiowanych według (4.35), można przekształcić wzór rozkładu szybkiego algorytmu dwuetapowego dla przekształcenia DST2D-II (4.33) tak, aby otrzymać żądany szybki algorytm z przerzedzeniem w czasie. Wówczas wzór rozkładu takiego algorytmu przyjmie postać wyrażenia

$$
\begin{aligned}
X_{N\cdot M}^{S2DII}(k,l) &= (X_4^{S2DII}(k,l) + X_5^{S2DII}(k,l) + \\
&+X_6^{S2DII}(k,l) + X_7^{S2DII}(k,l))C_N^{k+1}C_M^{l+1} - \\
&-(X_4^{S2DII}(\frac{N}{2}-k-2,l) - X_5^{S2DII}(\frac{N}{2}-k-2,l) + \\
&+X_6^{S2DII}(\frac{N}{2}-k-2,l) - X_7^{S2DII}(\frac{N}{2}-k-2,l))S_N^{k+1}C_M^{l+1} - \\
&-(X_4^{S2DII}(k,\frac{M}{2}-l-2) + X_5^{S2DII}(k,\frac{M}{2}-l-2) - \\
&-X_6^{S2DII}(k,\frac{M}{2}-l-2) - X_7^{S2DII}(k,\frac{M}{2}-l-2))C_N^{k+1}S_M^{l+1} + \\
&+(X_4^{S2DII}(\frac{N}{2}-k-2,\frac{N}{2}-l-2) - X_5^{S2DII}(\frac{N}{2}-k-2,\frac{M}{2}-l-2) - \\
&-X_6^{S2DII}(\frac{N}{2}-k-2,\frac{N}{2}-l-2) + X_7^{S2DII}(\frac{N}{2}-k-2,\frac{M}{2}-l-2))S_N^{k+1}S_M^{l+1}
\end{aligned}
\tag{4.40}
$$

dla $k = 0, 1, ..., N/2-2$ i $l = 0, 1, ..., M/2-2$, gdzie

$$
C_N^{k+1} = cos\left(\frac{\pi}{2N}(k+1)\right),\ S_N^{k+1} = sin\left(\frac{\pi}{2N}(k+1)\right),
$$

$$
C_M^{l+1} = cos\left(\frac{\pi}{2M}(l+1)\right),\ S_M^{l+1} = sin\left(\frac{\pi}{2M}(l+1)\right).
$$

Poniżej drobnym drukiem zamieszczono postać wygodną do programowej implementacji algorytmu FST2D-II z przerzedzeniem w czasie.

$$
\begin{aligned}
X_{N\cdot M}^{S2DII}(N-k-2,l) &= (X_4^{S2DII}(k,l)+X_5^{S2DII}(k,l)+ \\
&+ X_6^{S2DII}(k,l)+X_7^{S2DII}(k,l))S_N^{k+1}C_M^{l+1}+ \\
&+ (X_4^{S2DII}(\tfrac{N}{2}-k-2,l)-X_5^{S2DII}(\tfrac{N}{2}-k-2,l)+ \\
&+ X_6^{S2DII}(\tfrac{N}{2}-k-2,l)-X_7^{S2DII}(\tfrac{N}{2}-k-2,l))C_N^{k+1}C_M^{l+1}- \\
&- (X_4^{S2DII}(k,\tfrac{M}{2}-l-2)+X_5^{S2DII}(k,\tfrac{M}{2}-l-2)- \\
&- X_6^{S2DII}(k,\tfrac{M}{2}-l-2)-X_7^{S2DII}(k,\tfrac{M}{2}-l-2))S_N^{k+1}S_M^{l+1}- \\
&- (X_4^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{N}{2}-l-2)-X_5^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2)- \\
&- X_6^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{N}{2}-l-2)+X_7^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2))C_N^{k+1}S_M^{l+1}
\end{aligned}
$$

dla $k = 0, 1, ..., N/2 - 2$ i $l = 0, 1, ..., M/2 - 2$,

$$\begin{aligned}
X_{N\cdot M}^{S2DII}(k,M-l-2) &= (X_4^{S2DII}(k,l)+X_5^{S2DII}(k,l)+ \\
&+ X_6^{S2DII}(k,l)+X_7^{S2DII}(k,l))C_N^{k+1}S_M^{l+1}- \\
&- (X_4^{S2DII}(\tfrac{N}{2}-k-2,l)-X_5^{S2DII}(\tfrac{N}{2}-k-2,l)+ \\
&+ X_6^{S2DII}(\tfrac{N}{2}-k-2,l)-X_7^{S2DII}(\tfrac{N}{2}-k-2,l))S_N^{k+1}S_M^{l+1}+ \\
&+ (X_4^{S2DII}(k,\tfrac{M}{2}-l-2)+X_5^{S2DII}(k,\tfrac{M}{2}-l-2)- \\
&- X_6^{S2DII}(k,\tfrac{M}{2}-l-2)-X_7^{S2DII}(k,\tfrac{M}{2}-l-2))C_N^{k+1}C_M^{l+1}- \\
&- (X_4^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{N}{2}-l-2)-X_5^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2)- \\
&- X_6^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{N}{2}-l-2)+X_7^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2))S_N^{k+1}C_M^{l+1}
\end{aligned}$$

dla $k = 0, 1, ..., N/2 - 2$ i $l = 0, 1, ..., M/2 - 2$,

$$\begin{aligned}
X_{N\cdot M}^{S2DII}(N-k-2,M-l-2) &= (X_4^{S2DII}(k,l)+X_5^{S2DII}(k,l)+ \\
&+ X_6^{S2DII}(k,l)+X_7^{S2DII}(k,l))S_N^{k+1}S_M^{l+1}+ \\
&+ (X_4^{S2DII}(\tfrac{N}{2}-k-2,l)-X_5^{S2DII}(\tfrac{N}{2}-k-2,l)+ \\
&+ X_6^{S2DII}(\tfrac{N}{2}-k-2,l)-X_7^{S2DII}(\tfrac{N}{2}-k-2,l))C_N^{k+1}S_M^{l+1}+ \\
&+ (X_4^{S2DII}(k,\tfrac{M}{2}-l-2)+X_5^{S2DII}(k,\tfrac{M}{2}-l-2)- \\
&- X_6^{S2DII}(k,\tfrac{M}{2}-l-2)-X_7^{S2DII}(k,\tfrac{M}{2}-l-2))S_N^{k+1}C_M^{l+1}+ \\
&+ (X_4^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{N}{2}-l-2)-X_5^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2)- \\
&- X_6^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{N}{2}-l-2)+X_7^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-l-2))C_N^{k+1}C_M^{l+1}
\end{aligned}$$

dla $k = 0, 1, ..., N/2 - 2$ i $l = 0, 1, ..., M/2 - 2$ oraz

$$\begin{aligned}
X_{N\cdot M}^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1) &= \tfrac{1}{2}(X_4^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)+ \\
&+ X_5^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)+ \\
&+ X_6^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)+X_7^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)), \\
X_{N\cdot M}^{S2DII}(N-1,\tfrac{M}{2}-1) &= \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)- \\
&- X_5^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)+ \\
&+ X_6^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)-X_7^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)),
\end{aligned}$$

$$\begin{aligned}
X_{N\cdot M}^{S2DII}(\tfrac{N}{2}-1,M-1) &= \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)+ \\
&+ X_5^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)- \\
&- X_6^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)-X_7^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)), \\
X_{N\cdot M}^{S2DII}(N-1,M-1) &= (X_4^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)- \\
&- X_5^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)- \\
&- X_6^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1)+X_7^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-1))
\end{aligned}$$

oraz

$$
\begin{aligned}
X_{N\cdot M}^{S2DII}(k,\tfrac{M}{2}-1) &= \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(k,\tfrac{M}{2}-1)+X_5^{S2DII}(k,\tfrac{M}{2}-1)+\\
&+ X_6^{S2DII}(k,\tfrac{M}{2}-1)+X_7^{S2DII}(k,\tfrac{M}{2}-1))C_N^{k+1}-\\
&- \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)-\\
&- X_5^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)+\\
&+ X_6^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)-\\
&- X_7^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1))S_N^{k+1},\\
X_{N\cdot M}^{S2DII}(k,M-1) &= (X_4^{S2DII}(k,\tfrac{M}{2}-1)+X_5^{S2DII}(k,\tfrac{M}{2}-1)-\\
&- X_6^{S2DII}(k,\tfrac{M}{2}-1)-X_7^{S2DII}(k,\tfrac{M}{2}-1))C_N^{k+1}-\\
&- (X_4^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)-\\
&- X_5^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)-\\
&- X_6^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)+\\
&+ X_7^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1))S_N^{k+1},\\
X_{N\cdot M}^{S2DII}(N-k-2,\tfrac{M}{2}-1) &= \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(k,\tfrac{M}{2}-1)+X_5^{S2DII}(k,\tfrac{M}{2}-1)+\\
&+ X_6^{S2DII}(k,\tfrac{M}{2}-1)+X_7^{S2DII}(k,\tfrac{M}{2}-1))S_N^{k+1}+\\
&+ \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)-\\
&- X_5^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)+\\
&+ X_6^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)-\\
&- X_7^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1))C_N^{k+1},\\
X_{N\cdot M}^{S2DII}(N-k-2,M-1) &= (X_4^{S2DII}(k,\tfrac{M}{2}-1)+X_5^{S2DII}(k,\tfrac{M}{2}-1)-\\
&- X_6^{S2DII}(k,\tfrac{M}{2}-1)-X_7^{S2DII}(k,\tfrac{M}{2}-1))S_N^{k+1}+\\
&+ (X_4^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)-\\
&- X_5^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)-\\
&- X_6^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1)+\\
&+ X_7^{S2DII}(\tfrac{N}{2}-k-2,\tfrac{M}{2}-1))C_N^{k+1}
\end{aligned}
$$

dla $k = 0, 1, ..., N/2-2$ i

$$
\begin{aligned}
X_{N\cdot M}^{S2DII}(\tfrac{N}{2}-1,l) &= \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(\tfrac{N}{2}-1,l)+X_5^{S2DII}(\tfrac{N}{2}-1,l)+\\
&+ X_6^{S2DII}(\tfrac{N}{2}-1,l)+X_7^{S2DII}(\tfrac{N}{2}-1,l))C_M^{l+1}-\\
&- \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)+\\
&+ X_5^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)-\\
&- X_6^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)-\\
&- X_7^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2))S_M^{l+1},
\end{aligned}
$$

$$
\begin{aligned}
X_{N\cdot M}^{S2DII}(\tfrac{N}{2}-1,M-l-2) &= \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(\tfrac{N}{2}-1,l)+X_5^{S2DII}(\tfrac{N}{2}-1,l)+\\
&+ X_6^{S2DII}(\tfrac{N}{2}-1,l)+X_7^{S2DII}(\tfrac{N}{2}-1,l))S_M^{l+1}+\\
&+ \tfrac{\sqrt{2}}{2}(X_4^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)+\\
&+ X_5^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)-\\
&- X_6^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)-\\
&- X_7^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2))C_M^{l+1},\\
X_{N\cdot M}^{S2DII}(N-1,l) &= (X_4^{S2DII}(\tfrac{N}{2}-1,l)-X_5^{S2DII}(\tfrac{N}{2}-1,l)+\\
&+ X_6^{S2DII}(\tfrac{N}{2}-1,l)-X_7^{S2DII}(\tfrac{N}{2}-1,l))C_M^{l+1}-\\
&- (X_4^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)-\\
&- X_5^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)-\\
&- X_6^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)+\\
&+ X_7^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2))S_M^{l+1},\\
X_{N\cdot M}^{S2DII}(N-1,M-l-2) &= (X_4^{S2DII}(\tfrac{N}{2}-1,l)-X_5^{S2DII}(\tfrac{N}{2}-1,l)+\\
&+ X_6^{S2DII}(\tfrac{N}{2}-1,l)-X_7^{S2DII}(\tfrac{N}{2}-1,l))S_M^{l+1}+\\
&+ (X_4^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)-\\
&- X_5^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)-\\
&- X_6^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2)+\\
&+ X_7^{S2DII}(\tfrac{N}{2}-1,\tfrac{M}{2}-l-2))C_M^{l+1}
\end{aligned}
$$

dla $l = 0, 1, ..., M/2 - 2$.

Na podstawie wzoru rozkładu (4.40) można skonstruować szybki algorytm z przerzedzeniem w czasie dla dyskretnego dwuwymiarowego przekształcenia sinusowego drugiego rodzaju o rozmiarze N na M, gdzie N i M są potęgami dwóch. Wówczas elementy $x(n,m)$ wejściowej macierzy danych muszą być przemieszane zgodnie z kolejnością realizowaną przez procedurę A.10. Implementacje szybkiego algorytmu DST2D-II zamieszczono w dodatku A w postaci procedury A.13.

Ponieważ szybki algorytm z przerzedzeniem w czasie powstał na bazie szybkiego algorytmu dwuetapowego (4.33), jedynie poprzez wprowadzenie innego przemieszania elementów macierzy danych wejściowych, to charakteryzuje go ta sama złożoność obliczeniowa, co znany algorytm dwuetapowy. Dokładną liczbę operacji rzeczywistych dodawań $Do_{N\cdot M}^{S2DII}$ i mnożeń $Mn_{N\cdot M}^{S2DII}$ można zatem wyznaczyć za pomocą wyrażeń podanych w poprzedniej sekcji.

Dwuwymiarowe dyskretne przekształcenie sinusowe czwartego typu

Ostatnim z poszukiwanych szybkich algorytmów jest szybki algorytm z przerzedzeniem w czasie dla dyskretnego dwuwymiarowego przekształcenia sinusowego czwartego rodzaju. Również w tym przypadku, jako punkt wyjścia przyjmujemy wzór rozkładu szybkiego algorytmu dwuetapowego, który poprzez wprowadzenie ciągów (4.35) przekształcony zostaje do postaci wzoru rozkładu algorytmu z przerzedzeniem w czasie. W tym celu definiujemy pomocnicze przekształcenia $N/2$ na $M/2$ punktowe, które operują odpowiednio na ciągach $a(n,m)$, $b(n,m)$, $c(n,m)$ i $d(n,m)$ dla $n = 0, 1, ..., N/2-1$

i $m = 0, 1, ..., M/2 - 1$

$$X_4^{S2DIV}(k,l) = \\ = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} a(n,m) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right) \tag{4.41}$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$,

$$X_5^{S2DIV}(k,l) = \\ = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} b(n,m) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$,

$$X_6^{S2DIV}(k,l) = \\ = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} c(n,m) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$ i

$$X_7^{S2DIV}(k,l) = \\ = \sum_{n=0}^{N/2-1} \sum_{m=0}^{M/2-1} d(n,m) sin\left(\frac{\pi}{\left(\frac{N}{2}\right)}(k+\frac{1}{2})(n+\frac{1}{2})\right) sin\left(\frac{\pi}{\left(\frac{M}{2}\right)}(l+\frac{1}{2})(m+\frac{1}{2})\right)$$

dla $k = 0, 1, ..., N/2 - 1$ i $l = 0, 1, ..., M/2 - 1$. Następnie korzystając z wprowadzonych przekształceń, oraz stosując się do własności (4.37) ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ i $d(n,m)$, można przekształcić wzór rozkładu dwuetapowego szybkiego algorytmu dla przekształcenia DST2D-IV (4.34) do następującej postaci

$$X_{N\cdot M}^{S2DIV}(k,l) = (X_4^{S2DIV}(k,l) + X_5^{S2DIV}(k,l) + \\ + X_6^{S2DIV}(k,l) + X_7^{S2DIV}(k,l)) C_N^{k+\frac{1}{2}} C_M^{l+\frac{1}{2}} - \\ - (X_4^{S2DIV}(\frac{N}{2} - k - 1, l) - X_5^{S2DIV}(\frac{N}{2} - k - 1, l) + \\ + X_6^{S2DIV}(\frac{N}{2} - k - 1, l) - X_7^{S2DIV}(\frac{N}{2} - k - 1, l)) S_N^{k+\frac{1}{2}} C_M^{l+\frac{1}{2}} - \\ - (X_4^{S2DIV}(k, \frac{M}{2} - l - 1) + X_5^{S2DIV}(k, \frac{M}{2} - l - 1) - \tag{4.42}$$

$$-X_6^{S2DIV}(k,\frac{M}{2}-l-1)-X_7^{S2DIV}(k,\frac{M}{2}-l-1))C_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}+$$

$$+(X_4^{S2DIV}(\frac{N}{2}-k-1,\frac{N}{2}-l-1)-X_5^{S2DIV}(\frac{N}{2}-k-1,\frac{M}{2}-l-1)-$$

$$-X_6^{S2DIV}(\frac{N}{2}-k-1,\frac{N}{2}-l-1)+X_7^{S2DIV}(\frac{N}{2}-k-1,\frac{M}{2}-l-1))S_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}$$

dla $k=0,1,...,N/2-1$ i $l=-0,1,...,M/2-1$, gdzie

$$C_N^{k+\frac{1}{2}}=cos\left(\frac{\pi}{2N}(k+\frac{1}{2})\right),\ S_N^{k+\frac{1}{2}}=sin\left(\frac{\pi}{2N}(k+\frac{1}{2})\right),$$

$$C_M^{l+\frac{1}{2}}=cos\left(\frac{\pi}{2M}(l+\frac{1}{2})\right),\ S_M^{l+\frac{1}{2}}=sin\left(\frac{\pi}{2M}(l+\frac{1}{2})\right).$$

Rozpisując zależność (4.42) według symetrii przekształceń składowych otrzymujemy następujące zależności, które umożliwiają łatwą implementację procedur realizujących algorytm FST2D-IV z przerzedzeniem w czasie.

$$\begin{aligned}
X_{N\cdot M}^{S2DIV}(N-k-1,l) &= (X_4^{S2DIV}(k,l)+X_5^{S2DIV}(k,l)+\\
&+ X_6^{S2DIV}(k,l)+X_7^{S2DIV}(k,l))S_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}+\\
&+ (X_4^{S2DIV}(\tfrac{N}{2}-k-1,l)-X_5^{S2DIV}(\tfrac{N}{2}-k-1,l)+\\
&+ X_6^{S2DIV}(\tfrac{N}{2}-k-1,l)-X_7^{S2DIV}(\tfrac{N}{2}-k-1,l))C_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}-\\
&- (X_4^{S2DIV}(k,\tfrac{M}{2}-l-1)+X_5^{S2DIV}(k,\tfrac{M}{2}-l-1)-\\
&- X_6^{S2DIV}(k,\tfrac{M}{2}-l-1)-X_7^{S2DIV}(k,\tfrac{M}{2}-l-1))S_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}-\\
&- (X_4^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)-X_5^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)-\\
&- X_6^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)+X_7^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1))C_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}
\end{aligned}$$

dla $k=0,1,...,N/2-1$ i $l=-0,1,...,M/2-1$,

$$\begin{aligned}
X_{N\cdot M}^{S2DIV}(k,M-l-1) &= (X_4^{S2DIV}(k,l)+X_5^{S2DIV}(k,l)+\\
&+ X_6^{S2DIV}(k,l)+X_7^{S2DIV}(k,l))C_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}-\\
&- (X_4^{S2DIV}(\tfrac{N}{2}-k-1,l)-X_5^{S2DIV}(\tfrac{N}{2}-k-1,l)+\\
&+ X_6^{S2DIV}(\tfrac{N}{2}-k-1,l)-X_7^{S2DIV}(\tfrac{N}{2}-k-1,l))S_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}+\\
&+ (X_4^{S2DIV}(k,\tfrac{M}{2}-l-1)+X_5^{S2DIV}(k,\tfrac{M}{2}-l-1)-\\
&- X_6^{S2DIV}(k,\tfrac{M}{2}-l-1)-X_7^{S2DIV}(k,\tfrac{M}{2}-l-1))C_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}-\\
&- (X_4^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)-X_5^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1)-\\
&- X_6^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)+X_7^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1))S_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}
\end{aligned}$$

dla $k=0,1,...,N/2-1$ i $l=-0,1,...,M/2-1$ oraz

$$\begin{aligned}
X_{N\cdot M}^{S2DIV}(N-k-1,M-l-1) &= (X_4^{S2DIV}(k,l)+X_5^{S2DIV}(k,l)+\\
&+ X_6^{S2DIV}(k,l)+X_7^{S2DIV}(k,l))S_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}+\\
&+ (X_4^{S2DIV}(\tfrac{N}{2}-k-1,l)-X_5^{S2DIV}(\tfrac{N}{2}-k-1,l)+
\end{aligned}$$

$$
\begin{aligned}
&+ \quad X_6^{S2DIV}(\tfrac{N}{2}-k-1,l)-X_7^{S2DIV}(\tfrac{N}{2}-k-1,l))C_N^{k+\frac{1}{2}}S_M^{l+\frac{1}{2}}+\\
&+ \quad (X_4^{S2DIV}(k,\tfrac{M}{2}-l-1)+X_5^{S2DIV}(k,\tfrac{M}{2}-l-1)-\\
&- \quad X_6^{S2DIV}(k,\tfrac{M}{2}-l-1)-X_7^{S2DIV}(k,\tfrac{M}{2}-l-1))S_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}+\\
&+ \quad (X_4^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)-X_5^{S2DIV}(\tfrac{N}{2}-k-\mathbf{1},\tfrac{M}{2}-l-1)-\\
&- \quad X_6^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{N}{2}-l-1)+X_7^{S2DIV}(\tfrac{N}{2}-k-1,\tfrac{M}{2}-l-1))C_N^{k+\frac{1}{2}}C_M^{l+\frac{1}{2}}
\end{aligned}
$$

dla $k = 0, 1, ..., N/2-1$ i $l = -0, 1, ..., M/2-1$.

Zgodnie z wyrażeniem (4.42) N na M-punktowe przekształcenie obliczane jest na bazie czterech przekształceń $N/2$ na $M/2$-punktowych, które operują bezpośrednio na elementach $x(n,m)$ macierzy danych wejściowych. Rekurencyjne użycie tej zależności pozwala na szybkie obliczanie dwuwymiarowego przekształcenia sinusowego czwartego typu o wymiarze N na M, gdzie N i M są potęgami dwóch. Elementy wejściowej macierzy danych muszą być przemieszane z kolejnością wynikającą z doboru elementów dla ciągów $a(n,m)$, $b(n,m)$, $c(n,m)$ i $d(n,m)$. Procedura A.10 z dodatku A realizuje wspomniane przemieszanie. W rezultacie otrzymuje się szybki algorytm z przerzedzeniem w czasie dla przekształcenia DST2D-IV. W dodatku A zamieszczono także procedurę A.14, która jest implementacją rozważanego szybkiego algorytmu w języku C.

Ponieważ szybki algorytm z przerzedzeniem w czasie powstał na bazie szybkiego algorytmu dwuetapowego, jedynie poprzez wprowadzenie nowego przemieszania elementów $x(n,m)$ macierzy danych wejściowych, to charakteryzuje go ta sama złożoność obliczeniowa, co znany algorytm dwuetapowy. Stąd dla wyznaczenia dokładnej liczby operacji rzeczywistych dodawań $Do_{N\cdot M}^{S2DIV}$ oraz mnożeń $Mn_{N\cdot M}^{S2DIV}$ służą zależności przedstawione w poprzedniej sekcji.

W niniejszej sekcji przedstawiono wzory rozkładu szybkich algorytmów z przerzedzeniem w czasie dla wszystkich rozważanych dwuwymiarowych dyskretnych przekształceń trygonometrycznych. Algorytmy te wyprowadzono na bazie szybkich algorytmów dwuetapowych poprzez użycie innej niż $bit-reverse$ permutacji elementów macierzy danych wejściowych. Stąd algorytmy te charakteryzuje ta sama złożoność obliczeniowa co algorytmy dwuetapowe, tj. złożoność rzędu $\mathcal{O}(N^2 log_2 N)$. Ponadto struktury szybkich algorytmów z przerzedzeniem w czasie spełniają wymagania stawiane przez proponowane schematy szybkich algorytmów adaptacyjnego obliczania trygonometrycznych przekształceń całkowych.

4.3. Podsumowanie i wnioski

W niniejszym rozdziale zamieszczone zostały definicje szybkich algorytmów z przerzedzeniem w czasie dla dyskretnych jedno- i dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju. Znajomość wspomnianych algorytmów jest wymagana do budowy szybkich algorytmów adaptacyjnych.

Dla przypadku przekształceń jednowymiarowych szybkie algorytmy z przerzedzeniem w czasie wyprowadzone zostały na podstawie wzorów rozkładu znanych szybkich algorytmów dwuetapowych (patrz [156]), jedynie poprzez zastosowanie odmiennego przemieszania elementów wejściowych ciągów danych. Stąd algorytmy te charakteryzuje identyczna złożoność obliczeniowa rzędu $\mathcal{O}(N log_2 N)$ (patrz sekcja 4.1.3).

W przypadku przekształceń dwuwymiarowych charakterystyka proponowanych algorytmów adaptacyjnych uniemożliwia zastosowanie popularnego podejścia wierszowo-kolumnowego (patrz [154]), które polega na przykładaniu przekształceń jednowymiarowych odpowiednio do wierszy i do kolumn wejściowej macierzy próbek sygnału. Stąd także i tutaj, należało opracować szybkie algorytmy z przerzedzeniem w czasie dla rozważanych przekształceń dyskretnych. Algorytmy te wyprowadzono podobnie jak dla przekształceń jednowymiarowych, tj. na bazie dwuwymiarowych szybkich algorytmów dwuetapowych, poprzez wprowadzenie odmiennego od $bit-reverse$ przemieszania elementów macierzy danych wejściowych. Opisy tych algorytmów, podobnie jak algorytmów realizujących wymagane przemieszania, przedstawiono w sekcji 4.2.3.

Szybkie algorytmy z przerzedzeniem w czasie dla jedno- i dwuwymiarowych dyskretnych przekształceń kosinusowych i sinusowych zostały opublikowane odpowiednio w pracach [54] i [55].

Należy tutaj zwrócić uwagę na fakt, iż proponowane szybkie algorytmy z przerzedzeniem w czasie posiadają struktury symetryczne, przez co mogą być w łatwy sposób implementowane zarówno programowo jak i sprzętowo, w tym również według schematów równoległych i rozproszonych.

Dla jedno- i dwuwymiarowych dyskretnych przekształceń Fouriera szybkie algorytmy z przerzedzeniem w czasie stanowią jedne z algorytmów najczęściej wykorzystywanych w praktyce, jednakże z uwagi na ich ogromne znaczenie dla rozważań zawartych w dalszej części książki, wzory rozkładów dla tych algorytmów zostały przedstawione i krótko opisane odpowiednio w sekcjach 4.1.1 i 4.2.1.

W dodatku A zamieszczono implementacje w języku C procedur realizujących wszystkie opisane szybkie algorytmy z przerzedzeniem w czasie, a także wymagane przemieszania elementów wejściowych struktur danych.

W kolejnym rozdziale zamieszczone zostaną propozycje szybkich algorytmów adaptacyjnych dla jedno- i dwuwymiarowych przekształceń całkowych Fouriera, na których budowę pozwoliły opracowane szybkie algorytmy z przerzedzeniem w czasie, i w których struktury odpowiednio wkomponowano mechanizmy oceny błędów dla algorytmów adaptacyjnych, tj. algorytmów (3.1) i (3.2) w przypadku przekształceń jednowymiarowych, oraz (3.3) i (3.4) dla przekształceń dwuwymiarowych.

5

Szybkie algorytmy adaptacyjne

W rozdziale 3 opisano propozycje algorytmów adaptacyjnych dla całkowych jedno- i dwuwymiarowych przekształceń Fouriera. Algorytmy te, bazując na dyskretnych przekształceniach trygonometrycznych, a także korzystając z zaproponowanych* reguł oceny rzeczywistych błędów przybliżania wartości przekształceń całkowych poprzez przekształcenia dyskretne, pozwalały na automatyczny dobór takiej liczby próbek sygnału, która jest wystarczająca do obliczenia przekształcenia całkowego z zadaną dokładnością. Ocena błędu realizowana była na podstawie wartości dwóch przekształceń dyskretnych, odpowiednio N i $2N$-punktowych w przypadku jednowymiarowym, oraz N na M i $2N$ na $2M$-punktowych dla przekształceń dwuwymiarowych. Jednakże konstrukcja zaproponowanych algorytmów nie dawała możliwości wykorzystania obliczeń z etapów wcześniejszych do obliczania wartości przekształceń na kolejnych etapach adaptacji. Takie podejście znacznie wydłużało całkowity czas działania algorytmu i sprawiało, że zaproponowane algorytmy adaptacyjne są mało efektywne, a ich praktyczne zastosowania ograniczone.

Przyspieszenie obliczeń, a tym samym uzyskanie złożoności rzędu $\mathcal{O}(Nlog_2N)$ dla przekształceń jednowymiarowych oraz złożoności $\mathcal{O}(N^2log_2N)$ w przypadku dwuwymiarowym, możliwe jest poprzez wbudowanie mechanizmów służących do oceny błędu bezpośrednio w struktury szybkich algorytmów z przerzedzeniem w czasie (patrz rozdział 4). Co prawda, dodatkowe obliczenia związane z oceną błędu wydłużają czas działania szybkiego algorytmu (jak wykazały badania o około kilka do kilkudziesięciu procent w zależności do wymiaru przekształcenia i wartości parametrów wejściowych (patrz rozdział 7)), jednakże rząd złożoności obliczeniowej, charakterystyczny dla szybkich algorytmów obliczania dyskretnych przekształceń trygonometrycznych, pozostaje ten sam.

W ostatnich latach badania prowadzone nad ulepszaniem metod obliczania dyskretnych przekształceń trygonometrycznych w dużym stopniu koncentrowały się również na technikach adaptacyjnych. Dla przykładu w pracach [81, 82] znajdujemy propozycje algorytmów adaptacyjnych, które dopasowują funkcje bazowe przekształcenia do sygnału

*Dotychczas podano wyrażenia oceny błędu w metryce Czebyszewa. Wyrażenia przedstawione w innych popularnych metrykach zamieszczone zostały w rozdziale 6.

wejściowego według kryterium: *błąd rekonstrukcji-współczynnik kompresji*, przy czym dopasowanie przekształcenia odbywa się poprzez modyfikacje operacji motylkowych z zachowaniem ich ortonormalności. Wszystkie obliczenia realizowane są zgodnie ze strukturami algorytmów typu Cooley'a-Tukey'a. Przekształcenie to daje lepsze wyniki kompresji od przekształceń: DCT-II, DFT, przekształcenia Haara oraz przekształcenia Walsha-Hadamarda, co wykazano na przykładzie danych pomiarowych z zakresu fizyki nuklearnej. Z kolei w pracach [120, 132, 147] zaprezentowano grupę algorytmów tzw. SA (ang. *Shape-Adaptive*), które dostosowują się do kształtu transformowanego obszaru danych. Obszar ten nie musi być prostokątny. Takie podejście pozwoliło na lepsze dopasowanie przekształceń do fragmentów obrazów otrzymywanych na przykład w procesie segmentacji, i tym samym na uzyskanie wyższych współczynników kompresji. Algorytmy SA-DCT zastosowano między innymi w kompresji sekwencji wideo w standardzie MPEG-4 [51]. Równie dobrze znane są zagadnienia dotyczące przekształceń kosinusowych z adaptacyjną kwantyzacją wektorową [6, 69], a także adaptacyjnych filtrów kosinusowych LMS [10, 30].

Techniki adaptacyjne stanowią jedne z szeroko rozwijanych podejść w ramach współczesnej informatyki, między innymi z obszaru cyfrowego przetwarzania danych (np. filtracja adaptacyjna [113, 161]), czy też z obszaru sztucznej inteligencji [90, 114].

W niniejszym rozdziale zamieszczono opisy szybkich algorytmów adaptacyjnych dla jedno- i dwuwymiarowych całkowych przekształceń Fouriera, oraz sinusowego i kosinusowego przekształcenia Fouriera. Wzmiankowane algorytmy wykorzystują struktury szybkich algorytmów z przerzedzeniem w czasie dla jedno- i dwuwymiarowych dyskretnych przekształceń, odpowiednio przekształcenia Fouriera oraz kosinusowego i sinusowego przekształcenia drugiego i czwartego rodzaju.

Jako pierwszy opisany został przypadek jednowymiarowego przekształcenia Fouriera (patrz sekcja 5.1). Następnie przypadki jednowymiarowych przekształceń kosinusowych i sinusowych obliczanych za pomocą dyskretnych przekształceń drugiego i czwartego rodzaju (patrz sekcja 5.2). Ostatnie dwie sekcje 5.3 i 5.4 poświęcono przekształceniom dwuwymiarowych.

5.1. Szybki algorytm adaptacyjny jednowymiarowego przekształcenia Fouriera

Dobór próbek dla przekształcenia $X_{2N}(k)$ w kolejnych etapach działania algorytmu adaptacyjnego Fouriera (patrz algorytm 3.1) kontroluje reguła opisana zależnością

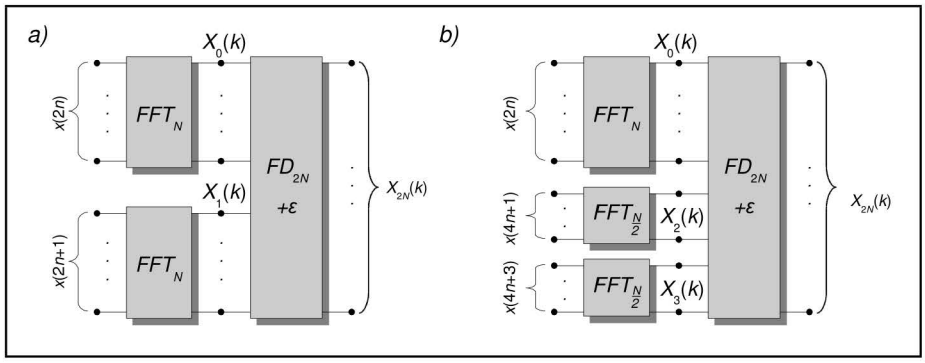
$$t_n = n\Delta t,$$

gdzie $\Delta t = T/(2N)$. Z kolei próbki, na zbiorze których obliczane było przekształcenie $X_N(k)$ na etapie poprzedzającym, można traktować jako próbki o indeksach parzystych dla etapu bieżącego, tj. etapu o $2N$ próbkach, ze względu na fakt, iż próbki te pobierane były w następujących chwilach czasowych

$$t_n = n\frac{T}{N} = 2n\Delta t$$

dla $n = 0, 1, ..., N-1$. Jeżeli przekształcenie $X_N(k)$ oznaczymy symbolicznie poprzez $X_0(k) = DFT_N\{x(2n)\}$, to na etapie bieżącym do obliczania przekształcenia $X_{2N}(k)$,

będącego $2N$-punktową kwadraturą numerycznego obliczania przekształcenia Fouriera, można wykorzystać dowolny z dwóch szybkich algorytmów przedstawionych w rozdziale 4, tj. szybki algorytm typu radix-2, lub typu "split-radix 2/4". Wówczas algorytm adaptacyjny powtarzać będzie strukturę wybranego szybkiego algorytmu z przerzedzeniem w czasie. Na poniższym rysunku pokazano schematy blokowe pojedynczych etapów szybkich algorytmów adaptacyjnych, konstruowanych w oparciu o szybkie algorytmy opisane zależnościami (4.2) (patrz rysunek 5.1a) i (4.4) (patrz rysunek 5.1b).



Rysunek 5.1: Schematy blokowe pojedynczych etapów dla szybkich algorytmów adaptacyjnych, budowanych w oparciu o szybkie algorytmy z przerzedzeniem w czasie typu radix-2 (a) i "split-radix 2/4" (b)

Bloki oznaczone symbolicznie przez $FD_{2N}+\epsilon$ (FD z ang. *forward decomposition*) realizują operacje wynikające z przyjętych wzorów rozkładu szybkich algorytmów, tj. operacje określone zależnościami (4.2) lub (4.4), oraz zawierają mechanizm automatycznego doboru takiej liczby próbek sygnału (dla metryki Czebyszewa jest to nierówność postaci (3.31)), która jest wystarczająca do obliczenia przekształcenia całkowego z zadaną dokładnością ϵ. Z kolei jako FFT oznaczono bloki szybkich algorytmów dla dyskretnego przekształcenia Fouriera, które operują na odpowiednio dobranych zbiorach próbek sygnału. W wyniku uzyskuje się schemat szybkiego algorytmu adaptacyjnego, który charakteryzuje złożoność obliczeniowa taka, jak dla wykorzystanych szybkich algorytmów z przerzedzeniem w czasie, tj. złożoność rzędu $\mathcal{O}(Nlog_2N)$. Oczywiście liczba N próbek musi spełniać warunek stawiany przez szybkie algorytmy z przerzedzeniem w czasie, tzn. musi być całkowitą potęgą liczby 2. Ze względu na prowadzoną na każdym etapie adaptacji kontrolę doboru liczby próbek sygnału, całkowita liczba operacji arytmetycznych powiększona zostaje o operacje związane z oceną błędu (np. te wynikające z obliczeń w ramach nierówności (3.31)). Złożoność tego procesu jest jednak rzędu $\mathcal{O}(N)$ (dla pojedynczego etapu). W rozdziale 7 zamieszczono wyniki eksperymentalnej analizy wpływu operacji potrzebnych do oceny błędu na całkowity czas wykonywania obliczeń.

Stąd szybki algorytm adaptacyjny dla przekształcenia Fouriera (AFFT), oparty na strukturach szybkich algorytmów z przerzedzeniem w czasie typu radix-2 lub "split-radix 2/4", można zapisać w następującej postaci

ALGORYTM 5.1 *(Szybki algorytm adaptacyjny dla przekształcenia Fouriera)*
Na wejściu dany musi być sygnał $x(t)$ określony na przedziale $[0,1]$ i przyjmujący wartości rzeczywiste, zbiór jego N próbek pobranych w punktach $t_n = n\Delta t$, gdzie N jest całkowitą potęgą 2 i $\Delta t = 1/N$, a także liczba $M \leqslant N$ współczynników widmowych, branych pod uwagę w procesie adaptacji, oraz dopuszczalna wartość błędu ϵ.

1. *Za pomocą szybkiego algorytmu FFT oblicz dyskretne N-punktowe przekształcenie $X_N(k) \equiv X_0(k)$.*
2. *Dobierz N próbek w punktach $t_{2n+1} = (2n+1)\Delta t/2$ dla $n = 0, 1, ..., N-1$.*
3. *W zależności od typu szybkiego algorytmu FFT (tj. radix-2 lub "split-radix 2/4") oblicz na podstawie dobranego zbioru próbek odpowiednio: N-punktowe przekształcenie $X_1(k) = DFT_N\{x((2n+1)\Delta t/2)\}$ lub dwa przekształcenia $N/2$-punktowe postaci $X_2(k) = DFT_{\frac{N}{2}}\{x((4n+1)\Delta t/4)\}$ i $X_3(k) = DFT_{\frac{N}{2}}\{x((4n+3)\Delta t/4)\}$.*
4. *Korzystając z wzoru (4.2) (dla algorytmu FFT typu radix-2) lub zależności (4.4) (dla algorytmu FFT typu "split-radix 2/4") oblicz przekształcenie $X_{2N}(k)$.*
5. *Wyznacz przybliżoną wartość ϵ^O_{MD} na podstawie następującej zależności $\epsilon^O_{MD} = \max\limits_{k=0,1,...,\frac{M}{2}} \left\{ \frac{1}{3} \| X_{2N}(k) - X_N(k) \| \right\}$. Jeżeli $\epsilon^O_{MD} \leqslant \epsilon$ to skocz do 7.*
6. *Podstaw $X_N(k) = X_{2N}(k)$ dla $k = 0, 1, ..., 2N-1$, $N = N \cdot 2$ oraz $\Delta t = \Delta t/2$. Skocz do 2.*
7. *Wypisz $X_{2N}(k)$ dla $k = 0, 1, ..., \frac{M}{2}$, $X_{2N}(2N-k)$ dla $k = 1, 2, ..., \frac{M}{2} - 1$, oraz wartości $2N$ i ϵ^O_{MD}.*

Koniec.

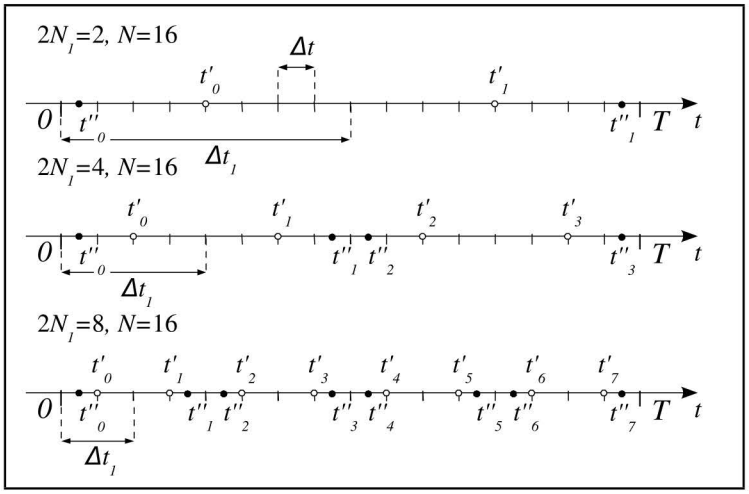
W dodatku A zamieszczono przykładową procedurę implementującą szybki algorytm adaptacyjny dla dyskretnego jednowymiarowego przekształcenia Fouriera, które obliczane jest według struktury szybkiego algorytmu z przerzedzeniem w czasie typu radix-2 (patrz procedura A.15).

5.2. Szybkie algorytmy adaptacyjne dla jednowymiarowych kosinusowych i sinusowych przekształceń Fouriera

Kolejny rozważany przypadek stanowią algorytmy adaptacyjne dla jednowymiarowych całkowych przekształceń kosinusowych i sinusowych Fouriera, które oblicza się numerycznie za pomocą dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju. Obowiązująca w przypadku stosowanych tutaj przekształceń dyskretnych reguła doboru próbek, które pozyskuje się w chwilach czasowych definiowanych według wyrażenia: $t'_n = (n+1/2)\Delta t$, uniemożliwia na kolejnych etapach adaptacji (patrz algorytm adaptacyjny 3.2) wykorzystanie do obliczania nowych przybliżeń wartości przekształcenia całkowego, obliczeń z etapów poprzedzających. Wynika to z faktu, iż zbiory próbek transformowanego sygnału, które pobierane są na sąsiednich etapach, będą rozłączne, tzn. punkty czasowe t'_n doboru próbek na wcześniejszym etapie, nie

będą należeć do zbioru punktów czasowych doboru próbek na etapie następnym. Stąd nie jest możliwe obliczanie wartości $2N$-punktowych kwadratur $X_{2N}^{PII}(k)$ $(X_{2N}^{PIV}(k))$ z wykorzystaniem wartości kwadratur $X_N^{PII}(k)$ $(X_N^{PIV}(k))$ o N punktach, gdzie P to typ przekształcenia. W rezultacie nie jest także możliwe bezpośrednie przyspieszanie obliczeń według schematu zaproponowanego dla przekształcenia Fouriera, i tym samym bezpośrednia konstrukcja analogicznych szybkich algorytmów adaptacyjnych.

W przypadku przekształceń kosinusowych i sinusowych Fouriera problem budowy szybkich algorytmów adaptacyjnych rozwiązuje się poprzez wprowadzenie wstępnego założenia pewnej maksymalnej, dostępnej na przedziale $[0, T]$ (w dalszej części przyjmuje się $T = 1$) liczby N próbek sygnału wejściowego. Dopiero wówczas, gdy znana jest taka liczba, możliwe jest zastosowanie odpowiedniego do typu przekształcenia szybkiego algorytmu z przerzedzeniem w czasie (patrz rozdział 4) i budowa w oparciu o jego strukturę szybkiego algorytmu adaptacyjnego. Takie podejście znajdzie zastosowanie tam, gdzie sygnał wejściowy najpierw jest nadpróbkowany, a następnie poszukuje się zbioru $N_1 < N$ próbek, które umożliwiają obliczanie reprezentacji sygnału w dziedzinie całkowego kosinusowego lub sinusowego przekształcenia Fouriera z pewną zadaną dokładnością ϵ. Jednakże wymagana przez szybkie algorytmy z przerzedzeniem w czasie permutacja wejściowego zbioru próbek sygnału (patrz permutacja p. w. cz (4.19)) sprawia, iż próbki te, chociaż ich zbiory zawierają się w kolejno rosnących etapach adaptacji, nie będą pobierane ani na początkach, ani w środkach przedziałów o długościach wynikających z rozważanej na danym etapie liczby próbek $2N_1$. Co więcej, reguła doboru próbek dla funkcji bazowych przekształceń dyskretnych, które wchodzą w skład struktur szybkich algorytmów z przerzedzeniem w czasie, odbiega w kolejnych etapach (poza etapem ostatnim, gdy $2N_1 = N$) od reguły stosowanej w procesie próbkowania sygnału wejściowego. Sytuację tę wyjaśnia rysunek 5.2. Na rysunku tym



Rysunek 5.2: Punkty czasowe doboru próbek funkcji bazowych dyskretnych przekształceń (patrz ∘) oraz transformowanego sygnału (patrz •) dla kroków $2N_1 = 2, 8, 4$ szybkiego algorytmu z przerzedzeniem w czasie, przy $N = 16$

symbolicznie poprzez • oznaczono punkty czasowe doboru próbek transformowanego sygnału w kolejnych etapach działania szybkiego algorytmu z przerzedzeniem w czasie,

których położenie opisuje poniższa zależność

$$t''n = (n + \frac{1}{2})\Delta t_1 - \frac{(-1)^n}{2}(\Delta t_1 - \Delta t)$$

dla $n = 0, 1, ..., 2N_1 - 1$, gdzie $\Delta t_1 = T/2N_1$ oraz $\Delta t = T/N$. Natomiast punkty oznaczone jako $\circ$ wyznaczają chwile czasowe $t'_n = (n + 1/2)\Delta t_1$ doboru próbek dla funkcji bazowych dyskretnych przekształceń (np. $cos(\pi/(2N_1)k(n + 1/2))$ dla DCT-II), które to chwile obowiązują w kolejnych krokach działania szybkiego algorytmu z przerzedzeniem w czasie, tutaj $2N_1 = 2, 4, 8$, przy czym $N = 16$. W konsekwencji wspomnianych faktów, uzyskiwane wyrażenia przybliżonego obliczania wartości przekształceń całkowych, nie będą spełniały założeń konstrukcyjnych wymaganych przez rozpatrywane w niniejszej książce równomierne złożone kwadratury prostokątów (3.8), oraz złożone kwadratury trapezów (3.10).

Stąd w dalszej części niniejszej sekcji dla potrzeb omówienia wyrażeń oceny błędu na kolejnych etapach adaptacji, przyjęte zostały odpowiednie założenia pozwalające na sprowadzenie przekształceń składowych szybkich algorytmów z przerzedzeniem w czasie do kwadratur postaci (3.8). Proponowany wariant szybkiego algorytmu adaptacyjnego dla kosinusowych i sinusowych przekształceń Fouriera opiera się na założeniu, iż próbki sygnału wejściowego $x(t)$ przyjmują w punktach t'_n i t''_n w przybliżeniu równe wartości, tzn. zachodzi związek $x(t'_n) \approx x(t''_n)$. Wówczas otrzymywane na kolejnych etapach kwadratury $X_{2N}^{PII}(k)$ można traktować jako równomierne kwadratury prostokątów o punktach leżących w środkach przedziałów dyskretyzacji, a proces ich obliczania realizować zgodnie ze wzorami rozkładu szybkich algorytmów z przerzedzeniem w czasie. To znaczy do obliczania wartości kwadratur $X_{2N}^{PII}(k)$ wykorzystywać obliczenia z etapu poprzedniego w postaci kwadratur $X_N^{PII}(k)$. Przy takich założeniach do oceny błędu numerycznego obliczania wartości przekształceń całkowych można stosować wyrażenia opracowane dla równomiernych kwadratur prostokątów, np. te podane w rozdziale 3 (patrz wzory (3.32) i (3.33)) dla metryki Czebyszewa, a także wzory oceny błędu w innych metrykach, które zestawione zostały w rozdziale 6.

Poniżej zamieszczono propozycję szybkiego algorytmu adaptacyjnego dla kosinusowych i sinusowych przekształceń Fouriera, który w świetle powyższych założeń skonstruowany został w oparciu o struktury szybkich algorytmów z przerzedzeniem w czasie. Algorytm ten w zależności od rodzaju przekształcenia oznaczać będziemy jako AFCT-II (AFCT-IV) dla przekształceń kosinusowych oraz AFST-II (AFST-IV) dla przekształceń sinusowych.

ALGORYTM 5.2 *(Szybki algorytm adaptacyjny dla kosinusowego i sinusowego przekształcenia Fouriera)*
Na wejściu dany musi być sygnał $x(t)$ określony na przedziale $[0, 1]$, zbiór jego N próbek $\{x(n) : n = 0, 1, ..., N - 1\}$ pobranych w punktach $t'_n = (n + 1/2)\Delta t$, gdzie $\Delta t = 1/N$, liczba $M \leqslant N$ współczynników widmowych uwzględnianych w procesie adaptacji, typ przekształcenia P, oraz dopuszczalna wartość błędu ϵ. Liczba N musi być całkowitą potęgą 2.

1. *Podstaw za N_1 najmniejszą potęgę 2, większą od M.*

2. *Posortuj wszystkie dostępne N próbek zgodnie z kolejnością p.w.cz.*

3. *Na zbiorze* $\{x(n) : n = 0, 1, ..., N_1 - 1\}$ *pierwszych* N_1 *próbek sygnału oblicz przekształcenie* $X_{N_1}^{PII}(k)$ $(X_{N_1}^{PIV}(k))$ *przy użyciu szybkiego algorytmu. Wówczas zachodzi tożsamościowa równość* $X_{N_1}^{PII}(k) \equiv X_2^{PII}(k)$ $(X_{N_1}^{PIV}(k) \equiv X_2^{PIV}(k))$.

4. *Operując na zbiorze* $\{x(n) : n = N_1, ..., 2N_1 - 1\}$ *kolejnych* N_1 *próbek oblicz przekształcenie* $X_3^{PII}(k)$ $(X_3^{PIV}(k))$ *przy pomocy z szybkiego algorytmu.*

5. *Korzystając z odpowiedniego wzoru rozkładu szybkiego algorytmu z przerzedzeniem w czasie (wzory (4.22) - (4.25)) oblicz przekształcenie* $X_{2N_1}^{PII}(k)$ $(X_{2N_1}^{PIV}(k))$.

6. *Oblicz przybliżoną wartość* ϵ_{MD}^{OPII} (ϵ_{MD}^{OPIV}) *błędu na podstawie następującej zależności*

$$\epsilon_{MD}^{OPII} = \max_{k=0,1,...,M-1} \left\{ \frac{1}{3p_k} \left| X_{2N_1}^{PII}(k) - X_{N_1}^{PII}(k) \right| \right\}$$

$$(\epsilon_{MD}^{OPIV} = \max_{k=0,1,...,M-1} \left\{ \frac{1}{3} \left| X_{2N_1}^{PIV}(k) - X_{N_1}^{PIV}(k) \right| \right\}).$$

   Jeżeli ϵ_{MD}^{OPII} $(\epsilon_{MD}^{OPIV}) \leqslant \epsilon$ *lub* $2N_1 = N$ *to skocz do 8.*

7. *Podstaw* $X_{N_1}^{PII}(k) = X_{2N_1}^{PII}(k)$ $(X_{N_1}^{PIV}(k) = X_{2N_1}^{PIV}(k))$ *dla* $k = 0, 1, ..., 2N_1 - 1$, $N_1 = N_1 \cdot 2$. *Skocz do 4.*

8. *Wypisz* $X_{2N_1}^{PII}(k)$ $(X_{2N_1}^{PII}(k))$ *dla* $k = 0, 1, ..., M - 1$, *oraz wartości* $2N_1$ *i* ϵ_{MD}^{OPII} (ϵ_{MD}^{OPIV}).

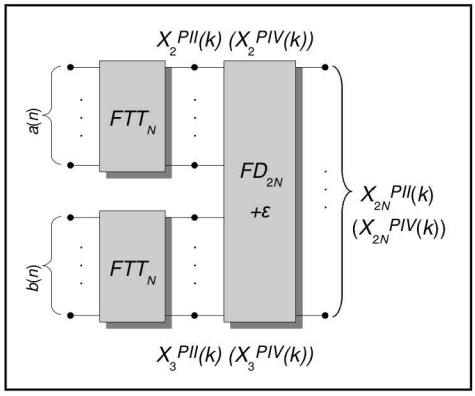
Koniec.

W dodatku A znaleźć można przykładową implementację powyższego algorytmu dla przypadku przekształcenia kosinusowego obliczanego numerycznie przy pomocy dyskretnego przekształcenia kosinusowego drugiego rodzaju (patrz procedura A.16). Ponieważ w pozostałych trzech przypadkach procedury te konstruuje się w sposób analogiczny, to nie zostały one zamieszczone w niniejszej monografii.

Na rysunku 5.3 przedstawiono schemat blokowy pojedynczego kroku dla szybkich algorytmów adaptacyjnego obliczania jednowymiarowych kosinusowych i sinusowych przekształceń Fouriera. Bloki $FD + \epsilon$ realizują odpowiednio piąty i szósty punkt algorytmu 5.2, tzn. zawierają operacje arytmetyczne opisane wzorami (4.22) - (4.25), oraz dokonują oceny błędu numerycznego przybliżenia wartości przekształceń całkowych przez kwadraturę $X_{2N_1}^{PII}(k)$ $(X_{2N_1}^{PII}(k))$. Z kolei jako FTT oznaczono bloki szybkich algorytmów obliczania dyskretnych przekształceń trygonometrycznych, tzn. w rozważanym przypadku przekształcenia kosinusowego lub sinusowego drugiego i czwartego rodzaju. Taki schemat obliczeniowy daje złożoność rzędu $\mathcal{O}(Nlog_2N)$.

5.3. Szybki algorytm adaptacyjny dla dwuwymiarowego przekształcenia Fouriera

W kolejnych dwóch sekcjach proponowane szybkie algorytmy adaptacyjne rozszerzone zostaną na przypadki dwuwymiarowych przekształceń całkowych. Jako pierwsze rozpatrzone zostanie całkowe przekształcenie Fouriera.



Rysunek 5.3: Schemat blokowy pojedynczego etapu szybkich algorytmów adaptacyjnych dla jednowymiarowych kosinusowych i sinusowych przekształceń Fouriera

Dla dwuwymiarowego całkowego przekształcenia Fouriera obliczanego w sposób numeryczny za pomocą przekształcenia dyskretnego opisanego wzorem (2.19), schemat doboru próbek cechujący szybki algorytm z przerzedzeniem w czasie, odpowiada schematowi wbudowanemu w przedstawiony w rozdziale trzecim algorytm adaptacyjny (patrz algorytm 3.3). Stąd budowa szybkiego algorytmu adaptacyjnego wymaga jedynie bezpośredniego przeniesienia reguł oceny błędu do struktury szybkiego algorytmu. Wówczas do obliczania kubatury $2N$ na $2M$-punktowej możliwe będzie wykorzystanie obliczeń z etapu poprzedniego, tj. wartości kubatury N na M-punktowej, uzyskując w konsekwencji redukcję obliczeń do poziomu rzędu $\mathcal{O}(N^2 log_2 N)$. Poniżej zamieszczono opis proponowanego szybkiego algorytmu adaptacyjnego dla dwuwymiarowego przekształcenia Fouriera (AFFT2D).

ALGORYTM 5.3 *(Szybki algorytm adaptacyjny dla dwuwymiarowego przekształcenia Fouriera)*
Na wejściu dany musi być sygnał $x(t,s)$ określony na przedziale $[0,1] \times [0,1]$ i przyjmujący wartości rzeczywiste, zbiór jego N na M próbek pobranych w punktach (t_n, s_m), gdzie $t_n = n\Delta t$ i $\Delta t = 1/N$, oraz $s_m = m\Delta s$ i $\Delta s = 1/M$, dopuszczalna wartość błędu ϵ, a także liczba współczynników widmowych $N_1 \leqslant N$ na $M_1 \leqslant M$, które brane są pod uwagę w procesie adaptacji.

1. *Na danym zbiorze próbek za pomocą szybkiego algorytmu oblicz $X^{2D}_{N\cdot M}(k,l)$, gdzie zachodzi tożsamościowa równość $X^{2D}_{N\cdot M}(k,l) \equiv X^{2D}_0(k,l)$.*

2. *Dobierz kolejno zbiory N na M próbek w nast. punktach: (t_{2n+1}, s_m), (t_n, s_{2m+1}) oraz (t_{2n+1}, s_{2m+1}) dla $n = 0, 1, ..., N-1$ i $m = 0, 1, ..., M-1$, gdzie punkty $t_{2n+1} = (2n+1)\Delta t/2$ i $s_{2m+1} = (2m+1)\Delta s/2$.*

3. *Dla tak dobranych zbiorów próbek oblicz przy pomocy szybkiego algorytmu następujące przekształcenia:*

$$X_1^{2D}(k,l) = DFT2D_{N\cdot M}\{x(t_{2n+1}, s_m))\},$$

$$X_2^{2D}(k,l) = DFT2D_{N\cdot M}\{x(t_n, s_{2m+1}))\},$$

$$X_3^{2D}(k,l) = DFT2D_{N\cdot M}\{x(t_{2n+1}, s_{2m+1}))\}.$$

4. *Na podstawie wzoru (4.27) rozkładu szybkiego algorytmu z przerzedzeniem w czasie oblicz przekształcenie* $X_{2N\cdot 2M}^{2D}(k,l)$.

5. *Oblicz przybliżoną wartość* ϵ_{MD}^{2DO} *na podstawie następującej zależności*

$$\epsilon_{MD}^{2DO}(k,l) = \max_B \left\{ \frac{1}{3} \| X_{2N\cdot 2M}^{2D}(k,l) - X_{N\cdot M}^{2D}(k',l') \| \right\}.$$

   Jeżeli $\epsilon_{MD}^{2DO} \leqslant \epsilon$ *to skocz do 7.*

6. *Podstaw* $X_{N\cdot M}^{2D}(k,l) = X_{2N\cdot 2M}^{2D}(k,l)$ *dla* $k = 0,1,...,2N-1$ *i* $l = 0,1,...,2M-1$, $N = N \cdot 2$, $M = M \cdot 2$ *oraz* $\Delta t = \Delta t/2$ *i* $\Delta s = \Delta s/2$. *Skocz do 2.*

7. *Wypisz:*

$$X_{2N\cdot 2M}^{2D}(k,l) \text{ dla } k = 0,1,...,\tfrac{N_1}{2} \text{ i } l = 0,1,...,\tfrac{M_1}{2},$$

$$X_{2N\cdot 2M}^{2D}(2N-k,l) \text{ dla } k = 1,2,...,\tfrac{N_1}{2}-1 \text{ i } l = 0,1,...,\tfrac{M_1}{2},$$

$$X_{2N\cdot 2M}^{2D}(k,2M-l) \text{ dla } k = 0,1,...,\tfrac{N_1}{2} \text{ i } l = 1,2,...,\tfrac{M_1}{2}-1,$$

$$X_{2N\cdot 2M}^{2D}(2N-k,2M-l) \text{ dla } k = 1,2,...,\tfrac{N_1}{2}-1 \text{ i } l = 1,2,...,\tfrac{M_1}{2}-1,$$

   oraz wartości $2N$, $2M$ *i* ϵ_{MD}^{2DO}.

Koniec.

Występujący powyżej symbol B określa zbiór czwórek (k,l,k',l'), które definiowane są według wyrażenia następującej postaci

$$B \triangleq \left\{ (k,l,k',l') : \begin{array}{c} k=0,1,...,\frac{N_1}{2}, l=0,1,...,\frac{M_1}{2}, k'=k \text{ i } l'=l \\ k=2N-(\frac{N_1}{2}-1),...,2N-1, l=0,1,...,\frac{M_1}{2}, k'=k-N \text{ i } l'=l \\ k=0,1,...,\frac{N_1}{2}, l=2M-(\frac{M_1}{2}-1),...,2M-1, k'=k \text{ i } l'=l-M \\ k=2N-(\frac{N_1}{2}-1),...,2N-1, l=2M-(\frac{M_1}{2}-1),...,2M-1, k'=k-N \text{ i } l'=l-M \end{array} \right\}.$$

5.4. Szybkie algorytmy adaptacyjne dla dwuwymiarowych kosinusowych i sinusowych przekształceń Fouriera

W ostatniej kolejności rozważone zostaną przypadki dwuwymiarowych kosinusowych i sinusowych całkowych przekształceń Fouriera. Dla tych przekształceń, podobnie jak w przypadku jednowymiarowym, opracowany został wariant szybkiego algorytmu adaptacyjnego. Algorytm ten opiera się na założeniu, iż wartości transformowanego sygnału $x(t,s)$ w dyskretnych chwilach czasowych (t''_n, s''_m) oraz (t'_n, s'_m) są w kolejnych krokach działania szybkiego algorytmu w przybliżeniu równe, tzn. zachodzi zawiązek $x(t'_n, s'_m) \approx x(t''_n, s''_m)$ dla $n = 0, 1, ..., 2N_1 - 1$ i $m = 0, 1, ..., 2M_1 - 1$, gdzie $2N_1$ i $2M_1$ to wymiar przekształcenia rozważanego w danym kroku, natomiast

$$t''_n = (n + \frac{1}{2})\Delta t_1 - \frac{(-1)^n}{2}(\Delta t_1 - \Delta t), t'_m = (n + \frac{1}{2})\Delta t_1,$$

$$s''_m = (m + \frac{1}{2})\Delta s_1 - \frac{(-1)^m}{2}(\Delta s_1 - \Delta s), s'_m = (m + \frac{1}{2})\Delta s_1$$

dla kroków dyskretyzacji $\Delta t_1 = T_1/2N_1$, $\Delta t = T_1/N$, $\Delta s_1 = T_2/2M_1$ i $\Delta s = T_2/M$ (w dalszej części rozdziału przyjmujemy $T_1 = T_2 = 1$). Przez N i M rozumiemy całkowitą liczbę próbek sygnału $x(t,s)$ rozłożonych na jego przedziale określoności i dobranych zgodnie z regułami doboru próbek dla dyskretnych przekształceń kosinusowych i sinusowych (patrz rozdział 2). Wówczas kubatury $X^{P2DII}_{2N_1 \cdot 2M_1}(k,l)$ ($X^{P2DIV}_{2N_1 \cdot 2M_1}(k,l)$) otrzymywane w kolejnych krokach działania szybkiego algorytmu z przerzedzeniem w czasie, można traktować jako równomierne kubatury złożone, opisywalne zależnością (3.23). Przy tak sformułowanych założeniach możliwe jest bezpośrednie zastosowanie do oszacowania błędu na kolejnych etapach adaptacji wyrażeń (3.39) lub (3.40), których dobór uzależniamy od typu przekształcenia.

Stąd możliwe jest sformułowanie proponowanego wariantu szybkiego algorytmu adaptacyjnego dla dwuwymiarowych kosinusowych i sinusowych przekształceń Fouriera. Algorytm ten w zależności od rodzaju przekształcenia oznaczać będziemy jako AFCT2D-II (AFCT2D-IV) dla przekształceń kosinusowych oraz AFST2D-II (AFST2D-IV) dla przekształceń sinusowych.

ALGORYTM 5.4 *(Szybki algorytm adaptacyjny dla dwuwymiarowych kosinusowych i sinusowych przekształceń Fouriera)*
Na wejściu dany musi być sygnał $x(t,s)$ określony na przedziale $[0,1] \times [0,1]$, dyskretny zbiór N na M próbek $\{x(n,m) : n = 0, 1, ..., N-1, m = 0, 1, ..., M-1\}$ pobranych w punktach (t'_n, s'_m), przy $t'_n = (n + 1/2)\Delta t$ i $s'_m = (m + 1/2)\Delta s$, gdzie $\Delta t = 1/N$ i $\Delta s = 1/M$, liczba $N_2 \leqslant N$ na $M_2 \leqslant M$ współczynników widmowych, które brane będą pod uwagę w procesie adaptacji, typ przekształcenia P, oraz dopuszczalna wartość błędu ϵ. Liczby N i M muszą być całkowitymi potęgami 2.

1. *Podstaw za N_1 najmniejszą potęgę 2, większą od N_2, podobnie za M_1 najmniejszą potęgę 2 i jednocześnie większą od M_2.*

2. *Posortuj wszystkie dostępne N na M próbek zgodnie z kolejnością p.w.cz. dla przekształceń dwuwymiarowych.*

3. *Na zbiorze* $\{x(n,m) : n = 0,1,...,N_1-1, m = 0,1,...,M_1-1\}$ *pierwszych* N_1 *na* M_1 *próbek sygnału oblicz przekształcenie* $X^{P2DII}_{N_1\cdot M_1}(k,l)$ $(X^{P2DIV}_{N_1\cdot M_1}(k,l))$ *przy użyciu szybkiego algorytmu. Zachodzi wówczas tożsamościowa równość postaci* $X^{P2DII}_{N_1\cdot M_1}(k,l) \equiv X^{P2DII}_{4}(k,l)$ $(X^{P2DIV}_{N_1\cdot M_1}(k) \equiv X^{P2DIV}_{4}(k,l))$.

4. *Operując na kolejnych zbiorach próbek postaci:*

$$\{x(n,m) : n = N_1,...,2N_1-1, m = 0,1,...,M_1-1\},$$

$$\{x(n,m) : n = 0,1,...,N_1-1, m = M_1,...,2M_1-1\},$$

$$\{x(n,m) : n = N_1,...,2N_1-1, m = M_1,...,2M_1-1\}$$

oblicz przy pomocy szybkich algorytmów następujące przekształcenia: $X^{P2DII}_{5}(k,l)$ $(X^{P2DIV}_{5}(k,l))$, $X^{P2DII}_{6}(k,l)$ $(X^{P2DIV}_{6}(k,l))$ *oraz* $X^{P2DII}_{7}(k,l)$ $(X^{P2DIV}_{7}(k,l))$.

5. *Korzystając z dobranego do typu przekształcenia wzoru rozkładu szybkiego algorytmu z przerzedzeniem w czasie (wzory (4.38) - (4.40), (4.42)) oblicz przekształcenie* $X^{P2DII}_{2N_1\cdot 2M_1}(k,l)$ $(X^{P2DIV}_{2N_1\cdot 2M_1}(k,l))$.

6. *Oblicz przybliżoną wartość* ϵ^{P2DOII}_{MD} (ϵ^{P2DOIV}_{MD}) *na podstawie następującej zależności*

$$\epsilon^{P2DOII}_{MD} = \max_{\substack{k=0,1,...,N_2-1 \\ l=0,1,...,M_2-1}} \left\{ \frac{1}{3p_k p_l} \left| X^{P2DII}_{2N_1\cdot 2M_1}(k,l) - X^{P2DII}_{N_1\cdot M_1}(k,l) \right| \right\}$$

$$(\epsilon^{P2DOIV}_{MD} = \max_{\substack{k=0,1,...,N_2-1 \\ l=0,1,...,M_2-1}} \left\{ \frac{1}{3} \left| X^{P2DIV}_{2N_1\cdot 2M_1}(k,l) - X^{P2DIV}_{N_1\cdot M_1}(k,l) \right| \right\}).$$

Jeżeli ϵ^{P2DOII}_{MD} $(\epsilon^{P2DOIV}_{MD}) \leqslant \epsilon$ *to skocz do 8.*

7. *Podstaw* $X^{P2DII}_{N_1\cdot M_1}(k,l) = X^{P2DII}_{2N_1\cdot 2M_1}(k,l)$ $(X^{P2DIV}_{N_1\cdot M_1}(k,l) = X^{P2DIV}_{2N_1\cdot 2M_1}(k,l))$ *dla* $k = 0,1,...,2N_1-1$ *i* $l = 0,1,...,2M_1-1$, *oraz* $N_1 = N_1\cdot 2$ *i* $M_1 = M_1\cdot 2$. *Skocz do 4.*

8. *Wypisz* $X^{P2DII}_{2N_1\cdot 2M_1}(k,l)$ $(X^{P2DII}_{2N_1\cdot 2M_1}(k,l))$ *dla* $k = 0,1,...,N_2-1$ *i* $l = 0,1,...,M_2-1$, *oraz wartości* $2N_1$ *na* $2M_1$ *i* ϵ^{P2DOII}_{MD} (ϵ^{P2DOIV}_{MD}).

Koniec.

Ze względu na fakt, iż implementacje programowe szybkich algorytmów adaptacyjnych 5.3 i 5.4, w odniesieniu do przypadku jednowymiarowego, realizuje się w sposób analogiczny z dokładnością do wymiaru przekształcenia, to przykłady takich implementacji nie zostały zamieszczone w dodatku A.

5.5. Podsumowanie i wnioski

W niniejszym rozdziale opisano propozycje szybkich algorytmów adaptacyjnych dla rozważanych całkowych przekształceń trygonometrycznych.

W przypadku jedno- i dwuwymiarowych przekształceń Fouriera mechanizm doboru próbek w kolejnych etapach adaptacji powtarza kolejność doboru próbek dla szybkich algorytmów z przerzedzeniem w czasie. Zatem tutaj konstrukcja szybkich algorytmów adaptacyjnych była możliwa poprzez bezpośrednie wbudowanie mechanizmu oceny błędu w struktury szybkich algorytmów. Takie rozwiązanie pozwoliło na redukcję obliczeń poprzez wykorzystanie do wyznaczania wartości kwadratur (kubatur) na kolejnych etapach, wartości kwadratur (kubatur) obliczanych na etapach poprzedzających. Stąd proponowane szybkie algorytmy adaptacyjne dla przekształceń Fouriera charakteryzuje złożoność obliczeniowa rzędu $\mathcal{O}(Nlog_2N)$ dla przypadku jednowymiarowego, oraz $\mathcal{O}(N^2log_2N)$ dla przekształceń dwuwymiarowych.

Dla trygonometrycznych przekształceń kosinusowych i sinusowych Fouriera budowa szybkich algorytmów adaptacyjnych wymagała przyjęcia pewnych dodatkowych założeń, co do sposobu zachowania transformowanego sygnału w otoczeniu punktów leżących w środkach podprzedziałów całkowania. Zabieg taki wymuszała inna kolejność doboru próbek dla algorytmów adaptacyjnych (patrz rozdział 3) i szybkich algorytmów z przerzedzeniem w czasie (patrz rozdział 4). W konsekwencji możliwa była konstrukcja szybkich algorytmów adaptacyjnych, których funkcjonalność oparto o równomierne kwadratury prostokątów (odpowiadające im kubatury całkowania numerycznego (patrz sekcja (5.4)), a schemat realizacji obliczeń powtarzał struktury szybkich algorytmów z przerzedzeniem w czasie o złożoności rzędu $\mathcal{O}(Nlog_2N)$ dla przekształceń jednowymiarowych, oraz odpowiednio $\mathcal{O}(N^2log_2N)$ dla przypadku dwuwymiarowego.

Zatem można jednoznacznie stwierdzić, iż dla wszystkich rozważanych przypadków przekształceń trygonometrycznych możliwa jest budowa szybkich algorytmów adaptacyjnych. Ich skuteczność poddana została praktycznej weryfikacji, a uzyskane wyniki eksperymentalne zestawiono w rozdziale 7. Z kolei w rozdziale 6 zamieszczone zostały propozycje wyrażeń oceny błędu w innych, niż cytowana dotychczas metryka maksymalnej odległości (metryka Czebyszewa).

Opisane szybkie algorytmy adaptacyjne dla dyskretnych przekształceń trygonometrycznych opublikowane zostały w pracach [56, 98, 99, 100].

6

Wyrażenia oceny błędu w innych metrykach

W dotychczasowych rozważaniach do oceny całościowego błędu numerycznego obliczania przekształceń całkowych przy pomocy przekształceń dyskretnych, wykorzystywano wyrażenia (3.31)-(3.33) oraz (3.38)-(3.40), które konstruowane były w metryce Czebyszewa, zwanej również metryką maksymalnej odległości (MD) (z jęz. ang. *maximum difference*) [124]. Na podstawie opisanego w rozdziale 3 mechanizmu oceny błędu dla poszczególnych wartości parametru k (lub parametrów k i l w przypadku dwuwymiarowym), możliwe jest wskazanie wyrażeń oceny błędu całościowego w innych popularnych metrykach, takich jak: metryka błędu średniokwadratowego (MSE) (z jęz. ang. *mean square error*), czy też metryka szczytowego stosunku sygnału do szumu (PSNR) (z jęz. ang. *peak signal to noise ratio*) [124].

Stąd w niniejszym rozdziale opisano propozycje wyrażeń oceny błędów w metrykach MSE i PSNR dla wszystkich rozważanych przypadków jedno- i dwuwymiarowych przekształceń Fouriera oraz kosinusowych i sinusowych przekształceń Fouriera, które obliczane są za pomocą przekształceń dyskretnych Fouriera oraz przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju. Przy czym rozpatruje się tutaj zarówno metryki pozwalające na przedstawienie błędu w postaci bezwzględnej (dla metryk MSE i PSNR), jak i w postaci błędu względnego (dla metryk MD, MSE i PSNR), który dla metryki MD wyrażony jest w procentach wartości modułów współczynników widmowych, a w przypadku metryk MSE oraz PSNR w procentach ich energii.

W dalszej części rozdziału proponowane wyrażenia oceny błędu pogrupowano względem wymiaru przekształceń. W sekcji 6.1 przedstawiono wyrażenia dla oceny błędów względnych i bezwzględnych dla przekształceń jednowymiarowych, natomiast w sekcji 6.2 zamieszczono propozycje analogicznych wyrażeń dla przypadku dwuwymiarowego.

6.1. Wyrażenia oceny błędu dla przypadku przekształceń jednowymiarowych

Na podstawie definicji popularnych metryk oceny błędu MSE i PSNR [124], a także mając na względzie związki występujące pomiędzy przekształceniami całkowymi i rozważanymi typami przekształceń dyskretnych (patrz rozdział 2), można we wspomnia-

nych metrykach zapisać wyrażenia, które pozwalają na obliczenie rzeczywistych wartości całościowych błędów numerycznego obliczania przekształceń całkowych za pomocą $2N$-punktowych przekształceń dyskretnych. W dalszych rozważaniach uwzględnia się jedynie pewną liczbę M niskoczęstotliwościowych współczynników spektralnych. Jako pierwszy rozpatrzony zostanie przypadek przekształcenia Fouriera. W tym przypadku przyjmujemy, że M jest liczbą parzystą.

Dla metryki MSE wyrażenie umożliwiające wyznaczenie rzeczywistego błędu numerycznego obliczania całkowego przekształcenia Fouriera przy pomocy DFT przyjmie wówczas postać

$$\begin{aligned}\epsilon^R_{MSE} &\triangleq \frac{1}{M}\sum_{k=-(\frac{M}{2}-1)}^{\frac{M}{2}} \|X(\omega_k) - X_{2N}(k)\|^2 = \frac{1}{M}\sum_{k=1}^{\frac{M}{2}-1} \|X(\omega_{-k}) - X_{2N}(2N-k)\|^2 + \\ &+ \frac{1}{M}\sum_{k=0}^{\frac{M}{2}} \|X(\omega_k) - X_{2N}(k)\|^2 = \frac{1}{M}\left(\|X(\omega_0) - X_{2N}(0)\|^2 + \right. \\ &+ \left. \|X(\omega_{\frac{M}{2}}) - X_{2N}(\frac{M}{2})\|^2 + 2\sum_{k=1}^{\frac{M}{2}-1} \|X(\omega_k) - X_{2N}(k)\|^2\right),\end{aligned} \tag{6.1}$$

gdzie $\omega_k = k\Delta\omega$ i $\Delta\omega = 2\pi/T$ (patrz rozdział 2), przy czym przyjmujemy $T = 1$. Taki sposób rozpisania powyższego wyrażenia wynika bezpośrednio z zależności $X_{2N}(-k) = X_{2N}(2N-k)$ (patrz rozdział 2 i wzór (2.7c)), z własności symetrii widma amplitudowego przekształcenia DFT (patrz Własność 2.3.1), oraz z analogicznej własności symetrii: $\|X(\omega_{-k})\| = \|X(\omega_k)\|$, którą można łatwo wykazać dla przekształcenia Fouriera w postaci całkowej.

Zgodnie z rozważaniami zamieszczonymi w rozdziale 3 wiadomo, iż przybliżoną wartość błędu numerycznego obliczania $X(\omega_k)$ za pomocą kwadratury $X_{2N}(k)$, można wyznaczyć jako: $\|X_{2N}(k) - X_N(k)\|/3$. Stąd po podstawieniu tej zależności do wzoru (6.1) otrzymuje się szukane wyrażenie służące do oceny błędu całościowego, którego postać wyrażona zostaje w metryce MSE

$$\begin{aligned}\epsilon^O_{MSE} &\triangleq \frac{1}{9M}\left(\|X_{2N}(0) - X_N(0)\|^2 + \|X_{2N}(\frac{M}{2}) - X_N(\frac{M}{2})\|^2 + \right. \\ &+ \left. 2\sum_{k=1}^{\frac{M}{2}-1} \|X_{2N}(k) - X_N(k)\|^2\right) \approx \epsilon^R_{MSE}.\end{aligned} \tag{6.2}$$

Dla metryki PSNR rzeczywistą wartość błędu można obliczyć według zależności

$$\epsilon^R_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,\frac{M}{2}}\left\{\|X(\omega_k)\|^2\right\}}{\epsilon^R_{MSE}}\right).$$

Wówczas przyjmując, że wartości $\|X_{2N}(k)\|$ są bliskie modułom wartości przekształceń całkowych $\|X(\omega_k)\|$, to przybliżoną wartość powyższego wyrażenia można obliczyć

według następującej zależności

$$\epsilon^{O}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,\frac{M}{2}}\left\{\|X_{2N}(k)\|^2\right\}}{\epsilon^{O}_{MSE}}\right) \approx \epsilon^{R}_{PSNR}. \tag{6.3}$$

Tym samym w postaci wzorów (6.2) i (6.3) otrzymujemy wyrażenia oceny rzeczywistego błędu obliczania całkowego przekształcenia Fouriera, które skonstruowane są w bezwzględnych metrykach MSE oraz PSNR.

Dla przekształceń kosinusowych i sinusowych Fouriera, które obliczane są odpowiednio za pomocą $2N$-punktowych dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju, wzory umożliwiające obliczanie całościowych błędów rzeczywistych MSE oraz PSNR dla liczby M niskoczęstotliwościowych współczynników widmowych, zapisuje się jako

$$\epsilon^{RPII}_{MSE} \triangleq \frac{1}{M}\sum_{k=0}^{M-1}\left|X^{P}(\omega^{II}_k) - X^{PII}_{2N}(k)\right|^2,$$

$$\left(\epsilon^{RPIV}_{MSE} \triangleq \frac{1}{M}\sum_{k=0}^{M-1}\left|X^{P}(\omega^{IV}_k) - X^{PIV}_{2N}(k)\right|^2\right)$$

oraz

$$\epsilon^{RPII}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,M-1}\left\{\left|X^{P}(\omega^{II}_k)\right|^2\right\}}{\epsilon^{RPII}_{MSE}}\right),$$

$$\left(\epsilon^{RPIV}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,M-1}\left\{\left|X^{P}(\omega^{IV}_k)\right|^2\right\}}{\epsilon^{RPIV}_{MSE}}\right)\right),$$

gdzie $\omega^{II}_k = k\Delta\omega'$ i $\omega^{IV}_k = (k+\frac{1}{2})\Delta\omega'$, przy $\Delta\omega' = \pi/T$ (patrz rozdział 2), natomiast P oznacza typ przekształcenia. Wówczas stosując zabieg analogiczny do zastosowanego dla przekształcenia Fouriera, można zbudować wyrażenia oceny rzeczywistych błędów MSE i PSNR, które w zależności od typu i rodzaju (II lub IV) przekształcenia, przyjmują postaci wzorów

$$\epsilon^{OPII}_{MSE} \triangleq \frac{1}{9M}\sum_{k=0}^{M-1}\left|X^{PII}_{2N}(k) - X^{PII}_{N}(k)\right|^2 \approx \epsilon^{RPII}_{MSE}, \tag{6.4a}$$

$$\left(\epsilon^{OPIV}_{MSE} \triangleq \frac{1}{9M}\sum_{k=0}^{M-1}\left|X^{PIV}_{2N}(k) - X^{PIV}_{N}(k)\right|^2 \approx \epsilon^{RPIV}_{MSE}\right) \tag{6.4b}$$

dla metryki MSE, oraz odpowiednio

$$\epsilon^{OPII}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,M-1}\left\{\left|X^{PII}_{2N}(k)\right|^2\right\}}{\epsilon^{OPII}_{MSE}}\right) \approx \epsilon^{RPII}_{PSNR}, \tag{6.5a}$$

$$\left(\epsilon_{PSNR}^{OPIV} \triangleq 10 \log_{10} \left(\frac{\max\limits_{k=0,1,\ldots,M-1} \left\{ \left| X_{2N}^{PIV}(k) \right|^2 \right\}}{\epsilon_{MSE}^{OPIV}} \right) \approx \epsilon_{PSNR}^{RPIV} \right) \tag{6.5b}$$

dla metryki PSNR.

W dalszej części rozdziału zamieszczone zostały wzory umożliwiające ocenę błędów względnych we wszystkich rozpatrywanych metrykach, tzn. metryce MD, MSE oraz PSNR. Dla metryki MD wartość błędu względnego wyrażona jest w procentach amplitud współczynników widmowych, natomiast w przypadku metryk MSE i PSNR w procentach energii tych współczynników.

Dla przypadku przekształcenia Fouriera przyjmuje się, że M jest liczbą parzystą. Wówczas rzeczywistą wartość względną błędu numerycznego obliczania całkowego przekształcenia Fouriera za pomocą przekształcenia DFT, definiujemy w metryce MD jako

$$\bar{\epsilon}_{MD}^{R} \triangleq \max_{k=0,1,\ldots,\frac{M}{2}} \left\{ \frac{\|X(\omega_k) - X_{2N}(k)\|}{\|X(\omega_k)\|} \right\} \cdot 100\%. \tag{6.6}$$

Jeżeli przyjąć, że zachodzi dla poszczególnych k następująca zależność pomiędzy wartościami błędów rzeczywistych, a ich przybliżeniami

$$\|X(\omega_k) - X_{2N}(k)\| \leqslant \|X_{2N}(k) - X_N(k)\|/3, \tag{6.7}$$

to przekształcając lewą stronę powyższej nierówności w następujący sposób

$$\|X(\omega_k) - X_{2N}(k)\| = \|X_{2N}(k) - X(\omega_k)\| \geqslant \|X_{2N}(k)\| - \|X_N(k)\|$$

i podstawiając otrzymany wynik do wzoru (6.7)

$$\|X(\omega_k)\| \geqslant \|X_{2N}(k)\| - \|X_{2N}(k) - X_N(k)\|/3, \tag{6.8}$$

uzyskujemy w rezultacie oddolne oszacowanie wartości $\|X(\omega_k)\|$. Oszacowanie to pozwala na budowę wzoru, który umożliwia przybliżone obliczenie wartości błędu ϵ_{MD}^{R}. Wzór ten przyjmuje postać poniższego wyrażenia

$$\bar{\epsilon}_{MD}^{O} \triangleq \max_{k=0,1,\ldots,\frac{M}{2}} \left\{ \frac{\|X_{2N}(k) - X_N(k)\|/3}{D(k)} \right\} \cdot 100\% \approx \bar{\epsilon}_{MD}^{R}, \tag{6.9}$$

gdzie $D(k) = \|X_{2N}(k)\| - \|X_{2N}(k) - X_N(k)\|/3$.

W przypadku metryk MSE i PSNR wartość błędu względnego wyrażana jest w procentach energii współczynników widmowych, tzn. rzeczywiste wartości względne tych błędów obliczamy odpowiednio jako

$$\bar{\epsilon}_{MSE}^{R} \triangleq \left(\frac{\epsilon_{MSE}^{R}}{\|X(\omega_0)\|^2 + \|X(\omega_{\frac{M}{2}})\|^2 + 2 \sum\limits_{k=1}^{\frac{M}{2}-1} \|X(\omega_k)\|^2} \right) \cdot 100\%,$$

oraz

$$\bar{\epsilon}_{PSNR}^{R} \triangleq 10 \log_{10} \left(\frac{\max\limits_{k=0,1,\ldots,\frac{M}{2}} \left\{ \|X(\omega_k)\|^2 \right\}}{\bar{\epsilon}_{MSE}^{R} \cdot 10^{-2}} \right).$$

Znów korzystając z nierówności (6.8) można łatwo skonstruować wyrażenia umożliwiające przybliżone obliczanie wartości błędów $\bar{\epsilon}^R_{MSE}$ oraz $\bar{\epsilon}^R_{PSNR}$. Wyrażenia te przyjmują postaci następujących wzorów

$$\bar{\epsilon}^O_{MSE} \triangleq \left(\frac{\epsilon^O_{MSE}}{D(0)^2 + D(\frac{M}{2})^2 + 2\sum\limits_{k=1}^{\frac{M}{2}-1} D(k)^2} \right) \cdot 100\% \approx \bar{\epsilon}^R_{MSE} \tag{6.10}$$

dla metryki MSE, oraz

$$\bar{\epsilon}^O_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,\frac{M}{2}} \left\{ \|X_{2N}(k)\|^2 \right\}}{\bar{\epsilon}^O_{MSE} \cdot 10^{-2}} \right) \approx \bar{\epsilon}^R_{PSNR} \tag{6.11}$$

dla metryki PSNR.

W analogiczny sposób można skonstruować wyrażenia oceny rzeczywistych wartości całościowego błędu względnego dla przypadku kosinusowych i sinusowych przekształceń Fouriera. Definicje względnych błędów rzeczywistych w rozważanych metrykach przyjmują odpowiednio postaci

$$\bar{\epsilon}^{RPII}_{MD} \triangleq \max_{k=0,1,\ldots,M-1} \left\{ \frac{\left| X^P(\omega^{II}_k) - X^{PII}_{2N}(k) \right|}{|X^P(\omega^{II}_k)|} \right\} \cdot 100\%,$$

$$\left(\bar{\epsilon}^{RPIV}_{MD} \triangleq \max_{k=0,1,\ldots,M-1} \left\{ \frac{\left| X^P(\omega^{IV}_k) - X^{PIV}_{2N}(k) \right|}{|X^P(\omega^{IV}_k)|} \right\} \cdot 100\% \right)$$

dla metryki MD,

$$\bar{\epsilon}^{RPII}_{MSE} \triangleq \left(\frac{\epsilon^{RPII}_{MSE}}{\sum\limits_{k=0}^{M-1} \left| X^P(\omega^{II}_k) \right|^2} \right) \cdot 100\%, \quad \left(\bar{\epsilon}^{RPII}_{MSE} \triangleq \left(\frac{\epsilon^{RPII}_{MSE}}{\sum\limits_{k=0}^{M-1} \left| X^P(\omega^{II}_k) \right|^2} \right) \cdot 100\% \right)$$

dla metryki MSE, oraz

$$\bar{\epsilon}^{RPII}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,M-1} \left\{ \left| X^P(\omega^{II}_k) \right|^2 \right\}}{\bar{\epsilon}^{RII}_{MSE} \cdot 10^{-2}} \right),$$

$$\left(\bar{\epsilon}^{RPIV}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,M-1} \left\{ \left| X^P(\omega^{IV}_k) \right|^2 \right\}}{\bar{\epsilon}^{RIV}_{MSE} \cdot 10^{-2}} \right) \right)$$

dla metryki PSNR. Wówczas w zależności od typu P i rodzaju (II lub IV) przekształcenia, wyrażenia służące do oceny błędów rzeczywistych w tych metrykach można zapisać jako następujące wzory

$$\bar{\epsilon}^{OPII}_{MD} \triangleq \max_{k=0,1,\ldots,M-1} \left\{ \frac{\left| X^{PII}_{2N}(k) - X^{PII}_{N}(k) \right| / 3}{D^{PII}(k)} \right\} \cdot 100\% \approx \epsilon^{RPII}_{MD} \tag{6.12a}$$

$$\left(\bar{\epsilon}_{MD}^{OPIV} \triangleq \max_{k=0,1,\ldots,M-1}\left\{\frac{\left|X_{2N}^{PIV}(k) - X_{N}^{PIV}(k)\right|/3}{D^{PIV}(k)}\right\}\cdot 100\% \approx \epsilon_{MD}^{RPIV}\right) \tag{6.12b}$$

dla metryki MD, gdzie $D^{PII}(k) = \left|X_{2N}^{PII}(k)\right| - \left|X_{2N}^{PII}(k) - X_{N}^{PII}(k)\right|/3$ oraz odpowiednio $D^{PIV}(k) = \left|X_{2N}^{PIV}(k)\right| - \left|X_{2N}^{PIV}(k) - X_{N}^{PIV}(k)\right|/3$, a także

$$\bar{\epsilon}_{MSE}^{OPII} \triangleq \left(\frac{\epsilon_{MSE}^{OPII}}{\sum_{k=0}^{M-1} D^{PII}(k)^2}\right)\cdot 100\% \approx \bar{\epsilon}_{MSE}^{RPII} \tag{6.13a}$$

$$\left(\bar{\epsilon}_{MSE}^{OPIV} \triangleq \left(\frac{\epsilon_{MSE}^{OPIV}}{\sum_{k=0}^{M-1} D^{PIV}(k)^2}\right)\cdot 100\% \approx \bar{\epsilon}_{MSE}^{RPIV}\right) \tag{6.13b}$$

dla metryki MSE, oraz

$$\bar{\epsilon}_{PSNR}^{OPII} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,M-1}\left\{\left|X_{2N}^{PII}(k)\right|^2\right\}}{\bar{\epsilon}_{MSE}^{OPII}\cdot 10^{-2}}\right) \approx \bar{\epsilon}_{PSNR}^{RPII} \tag{6.14a}$$

$$\left(\bar{\epsilon}_{PSNR}^{OPIV} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,M-1}\left\{\left|X_{2N}^{PIV}(k)\right|^2\right\}}{\bar{\epsilon}_{MSE}^{OPIV}\cdot 10^{-2}}\right) \approx \bar{\epsilon}_{PSNR}^{RPIV}\right) \tag{6.14b}$$

dla metryki PSNR.

Przedstawione wyrażenia oceny błędów mogą być stosowane w miejscu wyrażeń oceny bezwzględnych błędów rzeczywistych MD w punktach: czwartym algorytmu (3.1) oraz piątym algorytmu (5.1) (tzn. wzory (6.2), (6.3) oraz (6.9)-(6.11)), a także w punkcie czwartym algorytmu (3.2) oraz w punkcie szóstym algorytmu (5.2) (tzn. wzory (6.4a, 6.4b), (6.5a, 6.5b) i (6.12a, 6.12b)-(6.14a, 6.14b)). Wówczas przez ϵ rozumiemy dopuszczalną wartość błędu wyrażoną w wybranej metryce.

6.2. Wyrażenia oceny błędu dla przypadku przekształceń dwuwymiarowych

Kolejny rozpatrywany przypadek stanowią wyrażenia oceny błędów w bezwzględnych i względnych metrykach: MD, MSE oraz PSNR dla przekształceń dwuwymiarowych. Na początku rozważmy przypadek dwuwymiarowego przekształcenia Fouriera, które obliczane jest za pomocą $2N$ na $2M$-punktowego przekształcenia DFT2D. Wartości błędów wyznaczane będą dla liczby N_1 na M_1 współczynników niskoczęstotliwościowych (przyjmuje się, że liczby N_1 i M_1 są parzyste). Wówczas bezwzględną i rzeczywistą

wartość błędu całościowego wyrażoną w metryce MSE, można wyznaczyć na podstawie poniższej zależności

$$\begin{aligned}\epsilon_{MSE}^{R2D} \triangleq \frac{1}{N_1 M_1}\Bigg(&\sum_{k=0}^{\frac{N_1}{2}}\sum_{l=0}^{\frac{M_1}{2}}\|X^{2D}(\omega_k,\Omega_l)-X_{2N\cdot 2M}^{2D}(k,l)\|^2+\\ &+\sum_{k=1}^{\frac{N_1}{2}-1}\sum_{l=0}^{\frac{M_1}{2}}\|X^{2D}(\omega_{-k},\Omega_l)-X_{2N\cdot 2M}^{2D}(2N-k,l)\|^2+\\ &+\sum_{k=0}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{M_1}{2}-1}\|X^{2D}(\omega_k,\Omega_{-l})-X_{2N\cdot 2M}^{2D}(k,2M-l)\|^2+\\ &+\sum_{k=1}^{\frac{N_1}{2}-1}\sum_{l=1}^{\frac{M_1}{2}-1}\|X^{2D}(\omega_{-k},\Omega_{-l})-X_{2N\cdot 2M}^{2D}(2N-k,2M-l)\|^2\Bigg),\end{aligned} \tag{6.15}$$

gdzie $\omega_k = k\Delta\omega$, $\omega_l = l\Delta\Omega$ dla $\Delta\omega = 2\pi/T_1$ oraz $\Delta\Omega = 2\pi/T_2$, przy czym przyjmujemy $T_1 = 1$ oraz $T_2 = 1$. Taka postać wzoru (6.15) wynika z faktu, iż

$$X_{2N\cdot 2M}^{2D}(-k,l) = X_{2N\cdot 2M}^{2D}(2N-k,l),\ X_{2N\cdot 2M}^{2D}(k,-l) = X_{2N\cdot 2M}^{2D}(k,2M-l),$$

oraz

$$X_{2N\cdot 2M}^{2D}(-k,-l) = X_{2N\cdot 2M}^{2D}(2N-k,2M-l).$$

Korzystając z rozważań zamieszczonych w rozdziale 3 wiadomo, iż przybliżoną wartość błędu numerycznego obliczania $X^{2D}(\omega_k,\Omega_l)$ przy pomocy kubatury $X_{2N\cdot 2M}^{2D}(k,l)$, można obliczyć jako: $\|X_{2N\cdot 2M}^{2D}(k,l) - X_{N\cdot M}^{2D}(k,l)\|/3$ dla każdej pary (k,l) (patrz rozdział 3). Wówczas podstawiając tę zależność do wzoru (6.15), otrzymujemy następującą postać wyrażenia oceny wartości błędu ϵ_{MSE}^{R2D}

$$\begin{aligned}\epsilon_{MSE}^{O2D} \triangleq \frac{1}{9N_1 M_1}\Bigg(&\sum_{k=0}^{\frac{N_1}{2}}\sum_{l=0}^{\frac{M_1}{2}}\|X_{2N\cdot 2M}^{2D}(k,l)-X_{N\cdot M}^{2D}(k,l)\|^2+\\ &+\sum_{k=1}^{\frac{N_1}{2}-1}\sum_{l=0}^{\frac{M_1}{2}}\|X_{2N\cdot 2M}^{2D}(2N-k,l)-X_{N\cdot M}^{2D}(N-k,l)\|^2+\\ &+\sum_{k=0}^{\frac{N_1}{2}}\sum_{l=1}^{\frac{M_1}{2}-1}\|X_{2N\cdot 2M}^{2D}(k,2M-l)-X_{N\cdot M}^{2D}(k,M-l)\|^2+\\ &+\sum_{k=1}^{\frac{N_1}{2}-1}\sum_{l=1}^{\frac{M_1}{2}-1}\|X_{2N\cdot 2M}^{2D}(2N-k,2M-l)-X_{N\cdot M}^{2D}(N-k,M-l)\|^2\Bigg)\approx\epsilon_{MSE}^{2D}.\end{aligned} \tag{6.16}$$

Z kolei na obliczanie rzeczywistych wartości błędu w metryce PSNR pozwala wyrażenie postaci

$$\epsilon_{PSNR}^{R2D} \triangleq 10\log_{10}\left(\frac{\max\limits_{A}\left\{\|X^{2D}(\omega_k,\Omega_l)\|^2\right\}}{\epsilon_{MSE}^{R2D}}\right),$$

gdzie A oznacza zbiór par indeksów (k,l) przyjmujących następujące wartości

$$A \triangleq \left\{(k,l) : k = -(\frac{N_1}{2}-1), ..., 0, 1, ..., \frac{N_1}{2} \text{ i } l = -(\frac{M_1}{2}-1), ..., 0, 1, ..., \frac{M_1}{2}\right\}.$$

Wówczas zakładając, że wartości modułów z $X^{2D}_{2N\cdot 2M}(k,l)$ są w przybliżeniu równe modułom z $X^{2D}(\omega_k, \Omega_l)$ dla każdej pary (k,l), to wzór umożliwiający obliczenie przybliżonej wartości błędu ϵ^{R2D}_{PSNR} można zapisać w postaci

$$\epsilon^{O2D}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{\bar{B}} \|X^{2D}_{2N\cdot 2M}(k,l)\|^2}{\epsilon^{O2D}_{MSE}}\right) \approx \epsilon^{R2D}_{PSNR}, \tag{6.17}$$

gdzie przez $\bar{B}$ oznaczamy zbiór wszystkich par indeksów (k,l), które reprezentują rozważany zbiór niskoczęstotliwościowych współczynników widmowych

$$\bar{B} \triangleq \left\{(k,l) : \begin{array}{c} k=0,1,...,\frac{N_1}{2} \text{ i } l=0,1,...,\frac{M_1}{2} \\ k=2N-(\frac{N_1}{2}-1),...,2N-1 \text{ i } l=0,1,...,\frac{M_1}{2} \\ k=0,1,...,\frac{N_1}{2} \text{ i } l=2M-(\frac{M_1}{2}-1),...,2M-1 \\ k=2N-(\frac{N_1}{2}-1),...,2N-1 \text{ i } l=2M-(\frac{M_1}{2}-1),...,2M-1 \end{array}\right\}.$$

Dla dwuwymiarowych kosinusowych i sinusowych przekształceń Fouriera wzory definiujące bezwzględne rzeczywiste błędy numerycznego obliczania wartości przekształceń całkowych, liczonych przy pomocy dyskretnych przekształceń $2N$ na $2M$-punktowych, dla liczby N_1 na M_1 współczynników niskoczęstotliwościowych, przyjmują w metrykach MSE oraz PSNR następujące postaci

$$\epsilon^{RP2DII}_{MSE} \triangleq \frac{1}{N_1 M_1}\left(\sum_{k=0}^{N_1-1}\sum_{k=0}^{M_1-1}\left|X^{P2D}(\omega^{II}_k, \Omega^{II}_l) - X^{PII2D}_{2N\cdot 2M}(k,l)\right|^2\right),$$

$$\left(\epsilon^{RP2DIV}_{MSE} \triangleq \frac{1}{N_1 M_1}\left(\sum_{k=0}^{N_1-1}\sum_{k=0}^{M_1-1}\left|X^{P2D}(\omega^{IV}_k, \Omega^{IV}_l) - X^{PIV2D}_{2N\cdot 2M}(k,l)\right|^2\right)\right)$$

dla metryki MSE, oraz

$$\epsilon^{RP2DII}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,...,N_1-1 \text{ i } l=0,1,...,M_1-1}\left\{\left|X^{P2DII}(\omega^{II}_k, \Omega^{II}_l)\right|^2\right\}}{\epsilon^{RP2DII}_{MSE}}\right),$$

$$\left(\epsilon^{RP2DIV}_{PSNR} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,...,N_1-1 \text{ i } l=0,1,...,M_1-1}\left\{\left|X^{P2DIV}(\omega^{IV}_k, \Omega^{IV}_l)\right|^2\right\}}{\epsilon^{RP2DIV}_{MSE}}\right)\right),$$

gdzie P to typ przekształcenia, $\omega^{II}_k = k\Delta\omega$, $\omega^{IV}_k = (k+\frac{1}{2})\Delta\omega$ oraz $\Omega^{II}_l = l\Delta\Omega$ i $\omega^{IV}_l = (l+\frac{1}{2})\Delta\Omega$ dla $\Delta\omega = \pi/T_1$ oraz $\Delta\Omega = \pi/T_2$, przy czym przyjmujemy $T_1 = 1$ i $T_2 = 1$. Wówczas mając na uwadze wskazany w rozdziale 3 sposób przybliżonego obliczania wartości błędów rzeczywistych dla każdej pary (k,l), a także zakładając, że

zachodzą następujące zależności: $\left|X^{P2D}(\omega_k^{II},\Omega_l^{II}) \approx X_{2N\cdot 2M}^{P2DII}(\omega_k^{II},\Omega_l^{II})\right|$ i odpowiednio $\left|X^{P2D}(\omega_k^{IV},\Omega_l^{IV}) \approx X_{2N\cdot 2M}^{P2DIV}(\omega_k^{IV},\Omega_l^{IV})\right|$, to do przybliżonego obliczania błędów w metrykach MSE oraz PSNR, mogą posłużyć poniższe zależności

$$\epsilon_{MSE}^{OP2DII} \triangleq \frac{1}{9N_1M_1}\left(\sum_{k=0}^{N_1-1}\sum_{k=0}^{M_1-1}\left|X_{2N\cdot 2M}^{P2DII}(k,l) - X_{N\cdot M}^{P2DII}(k,l)\right|^2\right) \approx \epsilon_{MSE}^{P2DII}, \quad (6.18a)$$

$$\left(\epsilon_{MSE}^{OP2DIV} \triangleq \frac{1}{9N_1M_1}\left(\sum_{k=0}^{N_1-1}\sum_{k=0}^{M_1-1}\left|X_{2N\cdot 2M}^{P2DIV}(k,l) - X_{N\cdot M}^{P2DIV}(k,l)\right|^2\right) \approx \epsilon_{MSE}^{P2DIV}\right) \quad (6.18b)$$

dla matryki MSE, oraz

$$\epsilon_{PSNR}^{OP2DII} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,N_1-1 \text{ i } l=0,1,\ldots,M_1-1}\left\{\left|X_{2N\cdot 2M}^{P2DII}(k,l)\right|^2\right\}}{\epsilon_{MSE}^{OP2DII}}\right) \approx \epsilon_{PSNR}^{R2DII}, \quad (6.19a)$$

$$\left(\epsilon_{PSNR}^{RP2DIV} \triangleq 10\log_{10}\left(\frac{\max\limits_{k=0,1,\ldots,N_1-1 \text{ i } l=0,1,\ldots,M_1-1}\left\{\left|X_{2N\cdot 2M}^{P2DIV}(k,l)\right|^2\right\}}{\epsilon_{MSE}^{OP2DIV}}\right) \approx \epsilon_{PSNR}^{R2DIV}\right) \quad (6.19b)$$

dla metryki PSNR.

W ostatniej kolejności zaprezentowane zostaną wyrażenia oceny błędu względnego w metrykach: MD, MSE oraz PSNR. Jako pierwszy rozważony zostanie przypadek przekształcenia Fouriera. Także i tutaj zakładamy, że liczby N_1 i M_1, które wyznaczają zbiór współczynników niskoczęstotliwościowych branych pod uwagę podczas obliczania wartości błędów, są liczbami parzystymi. Wyrażenie umożliwiające wyznaczenie rzeczywistej wartości błędu względnego w metryce MD, przyjmie wówczas postać następującej zależności

$$\bar{\epsilon}_{MD}^{R2D} \triangleq \max_A\left\{\frac{\|X^{2D}(\omega_k,\Omega_l) - X_{2N\cdot 2M}^{2D}(k,l)\|}{\|X^{2D}(\omega_k,\Omega_l)\|}\right\}\cdot 100\%, \quad (6.20)$$

gdzie wartość tego błędu wyrażona jest w procentach modułów współczynników widmowych $X^{2D}(\omega_k,\Omega_l)$. Wówczas stosując podobne przekształcenia do tych, pokazanych dla przypadku przekształceń jednowymiarowych (patrz wzory (6.7) i (6.8)), można podać oddolne oszacowanie wartości $\|X^{2D}(\omega_k,\Omega_l)\|$, jako

$$\|X^{2D}(\omega_k,\Omega_l)\| \geqslant \|X_{2N\cdot 2M}^{2D}(k,l)\| - \|X_{2N\cdot 2M}^{2D}(k,l) - X_{N\cdot M}^{2D}(k',l')\|/3. \quad (6.21)$$

Oznaczmy prawą stronę powyższej nierówności przez $D^{2D}(k,l,k',l')$. Po uwzględnieniu we wzorze (6.20) oszacowania (6.21), uzyskujemy wzór umożliwiający przybliżone obliczanie wartości błędu $\bar{\epsilon}_{MD}^{R2D}$

$$\bar{\epsilon}_{MD}^{O2D} \triangleq \max_B\left\{\frac{\|X_{2N\cdot 2M}^{2D}(k,l) - X_{N\cdot M}^{2D}(k',l')\|/3}{D^{2D}(k,l,k',l')}\right\}\cdot 100\% \approx \bar{\epsilon}_{MD}^{R2D}. \quad (6.22)$$

Względne błędy rzeczywiste MSE i PSNR, których wartości wyrażone są w procentach energii współczynników widmowych, dla przypadku przekształcenia Fouriera definiujemy za pomocą wzorów postaci

$$\bar{\epsilon}_{MSE}^{R2D} \triangleq \left(\frac{\epsilon_{MSE}^{R2D}}{\sum\limits_{k=-(\frac{N_1}{2}-1)}^{\frac{N_1}{2}} \sum\limits_{l=-(\frac{M_1}{2}-1)}^{\frac{M_1}{2}} \|X^{2D}(\omega_k, \Omega_l)\|^2} \right) \cdot 100\%,$$

dla metryki MSE, oraz

$$\bar{\epsilon}_{PSNR}^{R2D} \triangleq 10 \log_{10} \left(\frac{\max\limits_{A} \left\{ \|X^{2D}(\omega_k, \Omega_l)\|^2 \right\}}{\bar{\epsilon}_{MSE}^{R2D} \cdot 10^{-2}} \right).$$

dla metryki PSNR. Wówczas na podstawie oszacowania (6.21), oraz przy założeniu, że wartości modułów $\|X_{2N\cdot 2M}^{2D}(k,l)\|$ są bliskie $\|X^{2D}(\omega_k, \Omega_l)\|$, można wskazać wzory pozwalające na przybliżone obliczanie wartości rzeczywistych błędów względnych w metrykach MSE oraz PSNR. Wzory te przyjmą odpowiednio postaci

$$\begin{aligned} \bar{\epsilon}_{MSE}^{O2D} \triangleq & \left(\epsilon_{MSE}^{O2D} / \left(\sum_{k=0}^{\frac{N_1}{2}} \sum_{l=0}^{\frac{M_1}{2}} D^{2D}(k,l,k,l)^2 + \right. \right. \\ & + \sum_{k=1}^{\frac{N_1}{2}-1} \sum_{l=0}^{\frac{M_1}{2}} D^{2D}(2N-k,l,N-k,l)^2 + \sum_{k=0}^{\frac{N_1}{2}} \sum_{l=1}^{\frac{M_1}{2}-1} D^{2D}(k,2M-l,k,M-l)^2 + \\ & \left. \left. + \sum_{k=1}^{\frac{N_1}{2}-1} \sum_{l=1}^{\frac{M_1}{2}-1} D^{2D}(2N-k,2M-l,N-k,M-l)^2 \right) \right) \cdot 100\% \approx \bar{\epsilon}_{MSE}^{R2D}, \end{aligned} \tag{6.23}$$

oraz

$$\bar{\epsilon}_{PSNR}^{O2D} \triangleq 10 \log_{10} \left(\frac{\max\limits_{B} \left\{ \|X_{2N\cdot 2M}^{2D}(k,l)\|^2 \right\}}{\bar{\epsilon}_{MSE}^{O2D} \cdot 10^{-2}} \right) \approx \bar{\epsilon}_{PSNR}^{R2D}. \tag{6.24}$$

W analogiczny sposób konstruujemy wyrażenia oceny błędów względnych w metrykach: MD, MSE i PSNR dla kosinusowych i sinusowych przekształceń Fouriera. W tym przypadku błędy względne w rozpatrywanych metrykach definiujemy kolejno jako

$$\bar{\epsilon}_{MD}^{RP2DII} \triangleq \max_{\substack{k=0,1,\ldots,N_1-1 \\ l=0,1,\ldots,M_1-1}} \left\{ \frac{\left| X^{P2DII}(\omega_k^{II}, \Omega_l^{II}) - X_{2N\cdot 2M}^{P2DII}(k,l) \right|}{|X^{P2DII}(\omega_k^{II}, \Omega_l^{II})|} \right\} \cdot 100\%,$$

$$\left(\bar{\epsilon}_{MD}^{RP2DIV} \triangleq \max_{\substack{k=0,1,\ldots,N_1-1 \\ l=0,1,\ldots,M_1-1}} \left\{ \frac{\left| X^{P2DIV}(\omega_k^{IV}, \Omega_l^{IV}) - X_{2N\cdot 2M}^{P2DIV}(k,l) \right|}{\left| X^{P2DIV}(\omega_k^{IV}, \Omega_l^{IV}) \right|} \right\} \cdot 100\% \right)$$

dla metryki MD,

$$\bar{\epsilon}_{MSE}^{RP2DII} \triangleq \left(\frac{\epsilon_{MSE}^{RP2DII}}{\sum_{k=0}^{N_1-1} \sum_{l=0}^{M_1-1} \left| X^{P2DII}(\omega_k^{II}, \Omega_l^{II}) \right|^2} \right) \cdot 100\%,$$

$$\left(\bar{\epsilon}_{MSE}^{RP2DIV} \triangleq \left(\frac{\epsilon_{MSE}^{RP2DIV}}{\sum_{k=0}^{N_1-1} \sum_{l=0}^{M_1-1} \left| X^{P2DIV}(\omega_k^{IV}, \Omega_l^{IV}) \right|^2} \right) \cdot 100\% \right)$$

dla metryki MSE, oraz

$$\bar{\epsilon}_{PSNR}^{RP2DII} \triangleq 10 \log_{10} \left(\frac{\max\limits_{k=0,1,\ldots,N_1-1 \text{ i } l=0,1,\ldots,M_1-1} \left\{ \left| X^{P2DII}(\omega_k^{II}, \Omega_l^{II}) \right|^2 \right\}}{\bar{\epsilon}_{MSE}^{RP2DII} \cdot 10^{-2}} \right),$$

$$\left(\bar{\epsilon}_{PSNR}^{RP2DIV} \triangleq 10 \log_{10} \left(\frac{\max\limits_{k=0,1,\ldots,N_1-1 \text{ i } l=0,1,\ldots,M_1-1} \left\{ \left| X^{P2DIV}(\omega_k^{IV}, \Omega_l^{IV}) \right|^2 \right\}}{\bar{\epsilon}_{MSE}^{RP2DIV} \cdot 10^{-2}} \right) \right)$$

dla metryki PSNR, gdzie P to typ przekształcenia. Następnie wprowadzając wyrażenia pomocnicze postaci

$$D^{P2DII}(k,l) \triangleq \left| X_{2N\cdot 2M}^{P2DII}(k,l) \right| - \left| X_{2N\cdot 2M}^{P2DII}(k,l) - X_{N\cdot M}^{P2DII}(k,l) \right| /3,$$

$$D^{P2DIV}(k,l) \triangleq \left| X_{2N\cdot 2M}^{P2DIV}(k,l) \right| - \left| X_{2N\cdot 2M}^{P2DIV}(k,l) - X_{N\cdot M}^{P2DIV}(k,l) \right| /3,$$

oraz zakładając, że moduły $\left| X_{2N\cdot 2M}^{P2DII}(k,l) \right|$ $\left(\left| X_{2N\cdot 2M}^{P2DIV}(k,l) \right| \right)$ są bliskie wartościom modułów $\left| X^{P2DII}(\omega_k^{II}, \Omega_l^{II}) \right|$ $\left(\left| X^{P2DIV}(\omega_k^{IV}, \Omega_l^{IV}) \right| \right)$, to wzory przybliżonego obliczania błędów względnych dla tych przekształceń, przyjmą w rozważanych metrykach postaci

(6.25a)

$$\bar{\epsilon}_{MD}^{OP2DII} \triangleq \max_{\substack{k=0,1,\ldots,N_1-1 \\ l=0,1,\ldots,M_1-1}} \left\{ \frac{\left| X_{2N\cdot 2M}^{P2DII}(k,l) - X_{N\cdot M}^{P2DII}(k,l) \right| /3}{D^{P2DII}(k,l)} \right\} \cdot 100\% \approx \bar{\epsilon}_{MD}^{RP2DII},$$

(6.25b)

$$\left(\bar{\epsilon}_{MD}^{OP2DIV} \triangleq \max_{\substack{k=0,1,\ldots,N_1-1 \\ l=0,1,\ldots,M_1-1}} \left\{ \frac{\left| X_{2N\cdot 2M}^{P2DIV}(k,l) - X_{N\cdot M}^{P2DIV}(k,l) \right| / 3}{D^{P2DIV}(k,l)} \right\} \cdot 100\% \approx \bar{\epsilon}_{MD}^{RP2DIV} \right)$$

dla metryki MD,

$$\bar{\epsilon}_{MSE}^{OP2DII} \triangleq \left(\frac{\epsilon_{MSE}^{OP2DII}}{\sum_{k=0}^{N_1-1} \sum_{l=0}^{M_1-1} D^{P2DII}(k,l)^2} \right) \cdot 100\% \approx \bar{\epsilon}_{MSE}^{RP2DII}, \tag{6.26a}$$

$$\left(\bar{\epsilon}_{MSE}^{OP2DIV} \triangleq \left(\frac{\epsilon_{MSE}^{OP2DIV}}{\sum_{k=0}^{N_1-1} \sum_{l=0}^{M_1-1} D^{P2DIV}(k,l)^2} \right) \cdot 100\% \approx \bar{\epsilon}_{MSE}^{RP2DIV} \right) \tag{6.26b}$$

dla metryki MSE, oraz

$$\bar{\epsilon}_{PSNR}^{OP2DII} \triangleq 10 \log_{10} \left(\frac{\max_{k=0,1,\ldots,N_1-1 \text{ i } l=0,1,\ldots,M_1-1} \left\{ \left| X_{2N\cdot 2M}^{P2DII}(k,l) \right|^2 \right\}}{\bar{\epsilon}_{MSE}^{OP2DII} \cdot 10^{-2}} \right), \tag{6.27a}$$

$$\left(\bar{\epsilon}_{PSNR}^{OP2DIV} \triangleq 10 \log_{10} \left(\frac{\max_{k=0,1,\ldots,N_1-1 \text{ i } l=0,1,\ldots,M_1-1} \left\{ \left| X_{2N\cdot 2M}^{P2DIV}(k,l) \right|^2 \right\}}{\bar{\epsilon}_{MSE}^{OP2DIV} \cdot 10^{-2}} \right) \right) \tag{6.27b}$$

dla metryki PSNR.

Zaprezentowane wyrażenia oceny błędów względnych i bezwzględnych w metrykach: MS, MSE oraz PSNR, mogą być stosowane w miejscu wyrażenia oceny bezwzględnego błędu MD, tj. w punkcie czwartym algorytmu 3.3 i w punkcie piątym algorytmu 5.3 (wzory (6.16), (6.17) oraz (6.22)-(6.24)), a także w punkcie czwartym algorytmu 3.4 i w punkcie szóstym algorytmu 5.4 (patrz wzory (6.18a, 6.18b), (6.19a, 6.19b) oraz (6.25a, 6.25b)-(6.27a, 6.27b)).

6.3. Podsumowanie i wnioski

W rozdziale tym zamieszczono wyrażenia oceny rzeczywistych błędów obliczania jedno- i dwuwymiarowych przekształceń całkowych poprzez przekształcenia dyskretne. Opisane wyrażenia pozwalają na ocenę błędów we względnych i bezwzględnych metrykach: MD, MSE oraz PSNR. W przypadku błędów względnych, ich wartości dla metryki MD

wyrażone zostały w procentach wartości amplitud współczynników widmowych, natomiast dla metryk MSE i PSNR, w procentach energii tych współczynników. Wyrażenia te mogą być stosowane w zaproponowanych algorytmach adaptacyjnych, w miejscu stosowanych do tej pory wzorów oceny błędów bezwzględnych w metryce MD.

Skuteczność zaprezentowanych wyrażeń została poddana weryfikacji eksperymentalnej w rozdziale 7 niniejszej książki.

7

Wyniki badań eksperymentalnych

Na treść niniejszego rozdziału składają się wyniki badań eksperymentalnych, które zostały przeprowadzone w celu praktycznej weryfikacji skuteczności rozważanych szybkich algorytmów adaptacyjnych, wraz z wyrażeniami oceny błędów we względnych i bezwzględnych metrykach: MD, MSE oraz PSNR. Badania przeprowadzono z wykorzystaniem sygnałów modelowych o znanych postaciach analitycznych. Ze względu na znajomość analitycznych postaci sygnałów wejściowych, możliwe było obliczenie dokładnych reprezentacji tych sygnałów w bazach rozważanych przekształceń całkowych. To z kolei pozwoliło na wyznaczenie rzeczywistych wartości błędów numerycznego obliczania przekształceń całkowych poprzez dyskretne przekształcenia trygonometryczne, i w rezultacie umożliwiło weryfikację skuteczności zaproponowanych wyrażeń oceny błędów. Badania nad skutecznością szybkich algorytmów adaptacyjnych przeprowadzono zarówno dla przypadków przekształceń jedno-, jak i dwuwymiarowych. Dla kosinusowych i sinusowych przekształceń Fouriera wyniki uzyskane za pomocą szybkich algorytmów adaptacyjnych, zostały dodatkowo porównane z wynikami otrzymanymi przy użyciu algorytmów adaptacyjnych (3.2) (w przypadku przekształceń jednowymiarowych) oraz (3.4) (w przypadku przekształceń dwuwymiarowych).

Druga część badań dotyczyła analizy wpływu dodatkowych operacji arytmetycznych, które wymagane są przez szybkie algorytmy adaptacyjne do wyznaczenia przybliżonych wartości błędów na kolejnych etapach adaptacji, na całkowity czas realizacji obliczeń. Uzyskane wyniki porównano z czasami obliczania przekształceń dyskretnych za pomocą szybkich algorytmów z przerzedzeniem w czasie, których struktury posłużyły do budowy szybkich algorytmów adaptacyjnych.

Wyniki badań pogrupowane zostały względem wymiarów przekształceń. Stąd w sekcji (7.1) zamieszczono wyniki otrzymane dla szybkich algorytmów adaptacyjnego obliczania jednowymiarowego przekształcenia Fouriera oraz jednowymiarowych przekształceń kosinusowych i sinusowych Fouriera. Z kolei w sekcji (7.2) znaleźć można wyniki uzyskane za pomocą szybkich algorytmów adaptacyjnych dla przekształceń dwuwymiarowych.

7.1. Wyniki badań dla przekształceń jednowymiarowych

W badaniach wykorzystano sygnały modelowe w postaci następujących funkcji, które przyjmują wartości rzeczywiste: funkcja wykładnicza malejąca $x_1(t) = e^{-8t}$, funkcja trygonometryczna $x_2(t) = sin(\pi t)$ ($x_2(t) = cos(\pi t)$ dla przekształcenia sinusowego Fouriera), impuls trójkątny $x_3(t) = \Lambda(2t-1)$, impuls prostokątny $x_4(t) = \Pi(2t-1)$, gdzie $t \in [0,1]$, impuls trójkątny i impuls prostokątny definiuje się jako [136]

$$\Lambda(t) \triangleq \begin{cases} 0 & \text{dla} \quad |t| > 1, \\ 1-|t| & \text{dla} \quad |t| \leqslant 1, \end{cases}$$

oraz

$$\Pi(t) \triangleq \begin{cases} 0 & \text{dla} \quad |t| > \frac{1}{2}, \\ \frac{1}{2} & \text{dla} \quad |t| = \frac{1}{2}, \\ 1 & \text{dla} \quad |t| < \frac{1}{2}. \end{cases}$$

Na wykresach (7.1a), (7.1b), (7.1c) i (7.1d) z rysunku 7.1 przedstawiono postaci użytych sygnałów modelowych.

Dla wszystkich badanych przypadków szybkich dyskretnych przekształceń adaptacyjnych Fouriera oraz kosinusowych i sinusowych przekształceń drugiego i czwartego rodzaju, jako maksymalną liczbę próbek sygnału wejściowego $x(t)$ przyjęto $N = 2048$. Natomiast liczba współczynników spektralnych, które uwzględniane były w procesie adaptacji, we wszystkich rozważanych przypadkach była identyczna i wynosiła $N_1 = 32$.

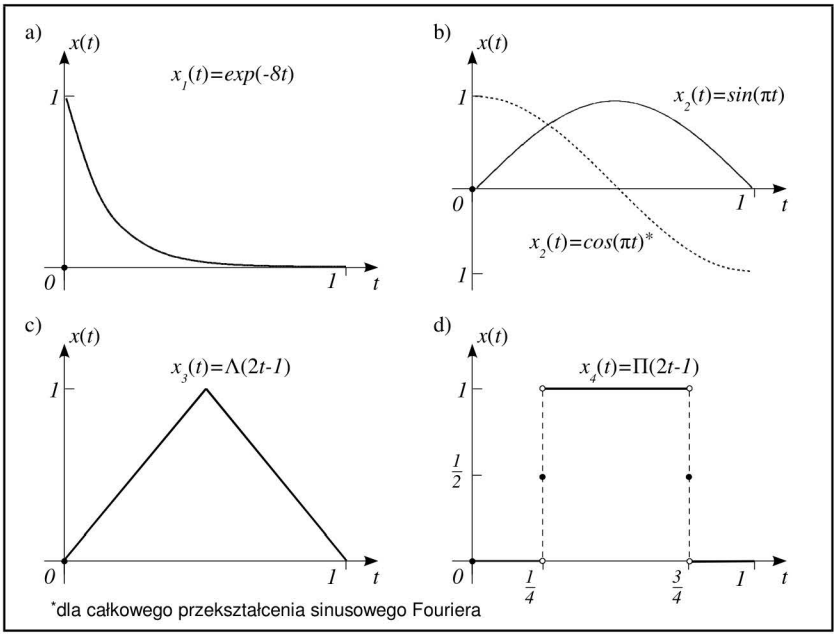
Uzyskane wyniki zestawiono w postaci wykresów oraz tabel. Na zamieszczonych wykresach przedstawiono stosunki przybliżonych wartości błędów do ich wartości rzeczywistych. Taki sposób reprezentacji wyników pozwolił na łatwą ocenę skuteczności reguł oceny błędów we wszystkich rozważanych metrykach błędów bezwzględnych i względnych: MD, MS oraz PSNR. Z kolei dokładne wyniki liczbowe zestawione zostały w tabelach i pogrupowane względem metryk dla błędów bezwzględnych i względnych.

Jako pierwsze zaprezentowano wyniki dla szybkiego algorytmu adaptacyjnego obliczania dyskretnego przekształcenia Fouriera.

Szybki algorytm adaptacyjny dla dyskretnego przekształcenia Fouriera
Dla sygnału wejściowego $x_1(t)$ całkowe przekształcenie Fouriera dla pulsacji ω_k, przy parametrze $k = 0, \pm 1, ..., \pm\infty$, przyjmuje następujące postaci

$$X_1(\omega_k) = \frac{e^{-8}-1}{2\pi^2 k^2 + 32}\left(-4 + i\pi k\right).$$

Znając postać całkowego przekształcenia Fouriera dla danego sygnału wejściowego, możliwe było wyznaczenie rzeczywistych wartości błędów bezwzględnych: ϵ^R_{MD}, ϵ^R_{MSE} oraz ϵ^R_{PSNR}, a także błędów względnych: $\bar{\epsilon}^R_{MD}$, $\bar{\epsilon}^R_{MSE}$ i $\bar{\epsilon}^R_{PSNR}$.



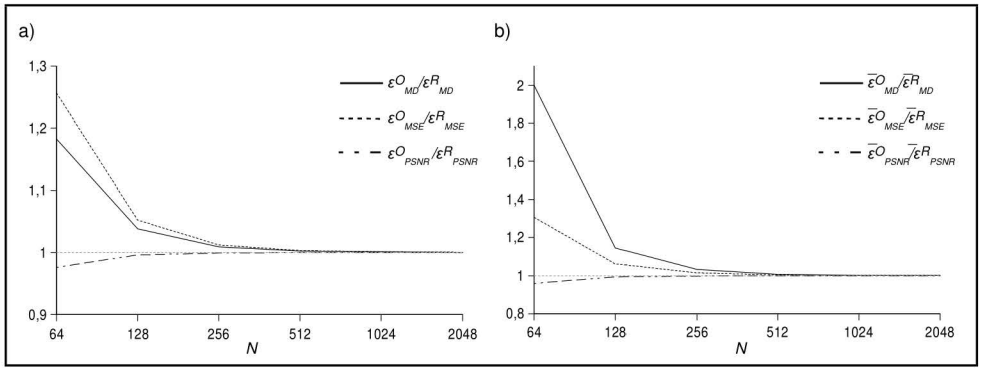
Rysunek 7.1: Sygnały modelowe wykorzystane podczas badań eksperymentalnych dla przypadku przekształceń jednowymiarowych: funkcja wykładnicza (a), funkcja trygonometryczna (b), impuls trójkątny (c) oraz impuls prostokątny (d)

Wyniki otrzymane na kolejnych etapach działania algorytmu AFFT (tj. wyniki uzyskane dla różnej liczby próbek sygnału wejściowego $N = 64, 128, 256, 1024$ oraz 2048) w postaci przybliżonych wartości błędów, które obliczano za pomocą wyrażeń zaproponowanych w rozdziale 6, porównano z wartościami rzeczywistymi i zestawiono na rysunku 7.2 oraz w tabelach 7.1 i 7.2.

Tabela 7.1: Wyniki oceny błędów bezwzględnych dla algorytmu AFFT i funkcji $x_1(t)$

N	$\epsilon^O_{MD} \cdot 10^5$	$\epsilon^R_{MD} \cdot 10^5$	$\epsilon^O_{MSE} \cdot 10^8$	$\epsilon^R_{MSE} \cdot 10^8$	$\epsilon^O_{PSNR}[dB]$	$\epsilon^R_{PSNR}[dB]$
64	236.1301	199.6417	188.1049	149.5972	39.2026	40.1861
128	50.3708	48.5301	9.4678	9.0015	52.1756	52.3921
256	12.1599	12.0504	0.5643	0.5574	64.4208	64.4733
512	3.0143	3.0075	0.0349	0.0348	76.5114	76.5244
1024	0.7520	0.7516	0.0022	0.0022	88.5649	88.5680
2048	0.1879	0.1879	0.0001	0.0001	100.6091	100.6097

Na podstawie otrzymanych wyników widać, iż przybliżone wartości błędów w początkowych etapach adaptacji (dla $N = 64$ i 128) w sposób dość zgrubny szacowały wartości błędów rzeczywistych (uzyskano stosunki błędów przybliżony/rzeczywisty nawet rzędu 2.0 patrz rysunek 7.2b), jednakże wraz ze wzrostem liczby próbek, wartości



Rysunek 7.2: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_1(t)$ i algorytmu AFFT: metryki bezwzględne (a), metryki względne (b)

Tabela 7.2: Wyniki oceny błędów względnych dla algorytmu AFFT i funkcji $x_1(t)$

N	$\bar{\epsilon}^O_{MD}$ [%]	$\bar{\epsilon}^R_{MD}$ [%]	$\bar{\epsilon}^O_{MSE} \cdot 10^6$ [%]	$\bar{\epsilon}^R_{MSE} \cdot 10^6$ [%]	$\bar{\epsilon}^O_{PSNR}$ [dB]	$\bar{\epsilon}^R_{PSNR}$ [dB]
64	37.7871	18.8898	3292.0706	2520.9987	26.7719	27.9196
128	5.2540	4.5918	161.0934	151.6923	39.8673	40.1257
256	1.1773	1.1402	9.5328	9.3939	52.1438	52.2068
512	0.2868	0.2846	0.5879	0.5858	64.2423	64.2579
1024	0.0713	0.0711	0.0366	0.0366	76.2977	76.3015
2048	0.0178	0.0178	0.0023	0.0023	88.3424	88.3432

przybliżone szybko zbliżały się do wartości błędów rzeczywistych. Co ważne, dla danego sygnału wejściowego, na wszystkich etapach adaptacji oszacowania błędów były oszacowaniami odgórnymi dla metryk: MD i MSE, oraz oszacowaniami oddolnymi dla metryki PSNR.

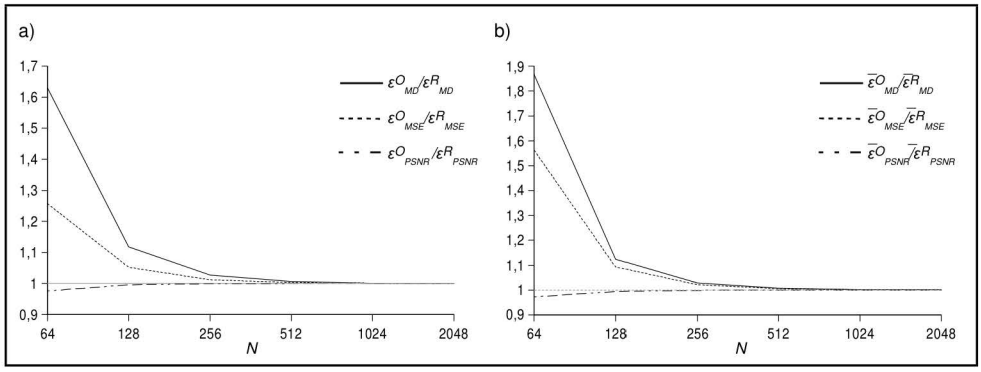
W przypadku drugiego sygnału modelowego $x_2(t) = sin(\pi t)$ postać widma całkowego przekształcenia Fouriera dla pulsacji ω_k można wyznaczyć za pomocą następującej zależności

$$X_2(\omega_k) = \begin{cases} \dfrac{1}{2} & \text{dla} \quad k = 0, \\ \dfrac{-2}{\pi^2 k^2} & \text{dla} \quad k \text{ nieparzystych}, \\ 0 & \text{dla} \quad k \text{ parzystych}, \end{cases}$$

gdzie parametr $k = 0, \pm 1, ..., \pm\infty$.

Wyniki uzyskane dla sygnału wejściowego $x_2(t)$ zestawiono w tabelach 7.3 i 7.4 oraz przedstawiono na rysunku 7.3.

W rozważanym przypadku otrzymano wyniki zbliżone do wyników uzyskanych dla sygnału modelowego $x_1(t)$. To znaczy, otrzymane oszacowania dla metryk MD i MSE



Rysunek 7.3: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_2(t)$ i algorytmu AFFT: metryki bezwzględne (a), metryki względne (b)

Tabela 7.3: Wyniki oceny błędów bezwzględnych dla algorytmu AFFT i funkcji $x_2(t)$

N	$\epsilon^O_{MD} \cdot 10^5$	$\epsilon^R_{MD} \cdot 10^5$	$\epsilon^O_{MSE} \cdot 10^{10}$	$\epsilon^R_{MSE} \cdot 10^{10}$	$\epsilon^O_{PSNR}[dB]$	$\epsilon^R_{PSNR}[dB]$
64	23.3003	14.2991	278.3271	178.3194	71.6303	73.5656
128	3.6716	3.2844	11.3930	10.4289	85.5108	85.8952
256	0.8267	0.8044	0.6552	0.6416	97.9136	98.0048
512	0.2014	0.2001	0.0402	0.0399	110.0403	110.0628
1024	0.0500	0.0500	0.0025	0.0025	122.1026	122.1082
2048	0.0125	0.0125	0.0002	0.0002	134.1491	134.1505

Tabela 7.4: Wyniki oceny błędów względnych dla algorytmu AFFT i funkcji $x_2(t)$

N	$\bar{\epsilon}^O_{MD}$ [%]	$\bar{\epsilon}^R_{MD}$ [%]	$\bar{\epsilon}^O_{MSE} \cdot 10^9$ [%]	$\bar{\epsilon}^R_{MSE} \cdot 10^9$ [%]	$\bar{\epsilon}^O_{PSNR}$ [dB]	$\bar{\epsilon}^R_{PSNR}$ [dB]
64	37.6948	20.1924	5570.2474	3566.4175	68.6171	70.5553
128	5.2133	4.6380	227.9000	208.5797	82.4997	82.8849
256	1.1677	1.1359	13.1053	12.8327	94.9030	94.9944
512	0.2845	0.2825	0.8031	0.7989	107.0299	107.0525
1024	0.0707	0.0705	0.0499	0.0499	111.0923	111.0979
2048	0.0176	0.0176	0.0031	0.0031	131.1387	131.1401

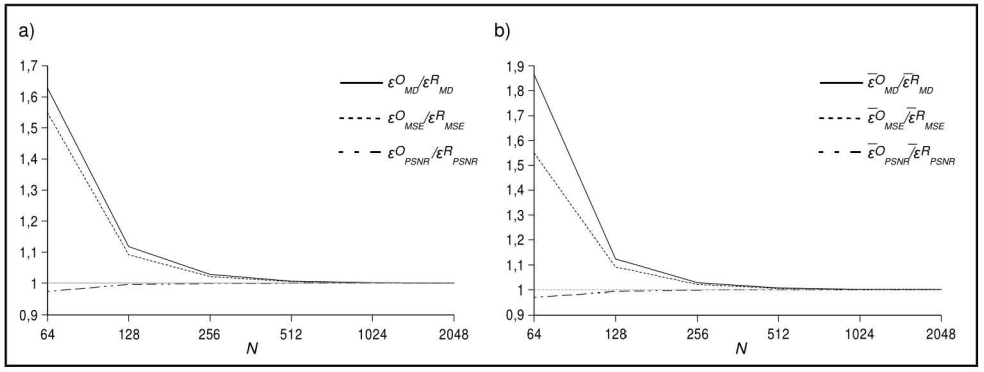
są oszacowaniami odgórnymi, natomiast dla metryki PSNR, uzsykano oszacowanie oddolne. Przybliżona wartość błędu dla wszystkich metryk w początkowych krokach adaptacji dla $N = 64$ i 128 daje oszacowanie dość zgrubne, natomiast wraz ze wzrostem liczby próbek sygnału wejściowego, szybko zbliża się do wartości rzeczywistej błędu.

Dla kolejnego sygnału modelowego $x_3(t)$ widmo całkowego przekształcenia Fouriera dla dyskretnych pulsacji ω_k, można w zależności od parametru $k = 0, \pm 1, ..., \pm\infty$

wyznaczyć z następującej zależności

$$X_3(\omega_k) = \frac{2}{\pi(1-4k^2)}.$$

Wyniki dla danego przypadku zaprezentowano na wykresach z rysunku 7.4 oraz zestawiono w postaci liczbowej w tabelach 7.5 oraz 7.6.



Rysunek 7.4: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_3(t)$ i algorytmu AFFT: metryki bezwzględne (a), metryki względne (b)

Tabela 7.5: Wyniki oceny błędów bezwzględnych dla algorytmu AFFT i funkcji $x_3(t)$

N	$\epsilon^O_{MD} \cdot 10^5$	$\epsilon^R_{MD} \cdot 10^5$	$\epsilon^O_{MSE} \cdot 10^{10}$	$\epsilon^R_{MSE} \cdot 10^{10}$	$\epsilon^O_{PSNR}[dB]$	$\epsilon^R_{PSNR}[dB]$
64	29.6462	18.2051	223.7917	144.4378	70.4810	72.3826
128	4.6745	4.1817	9.2265	8.4521	84.3290	84.7098
256	1.0525	1.0242	0.5310	0.5201	96.7284	96.8188
512	0.2565	0.2547	0.0325	0.0324	108.8544	108.8768
1024	0.0637	0.0636	0.0020	0.0020	120.9166	120.9221
2048	0.0159	0.0159	0.0001	0.0001	132.9630	132.9644

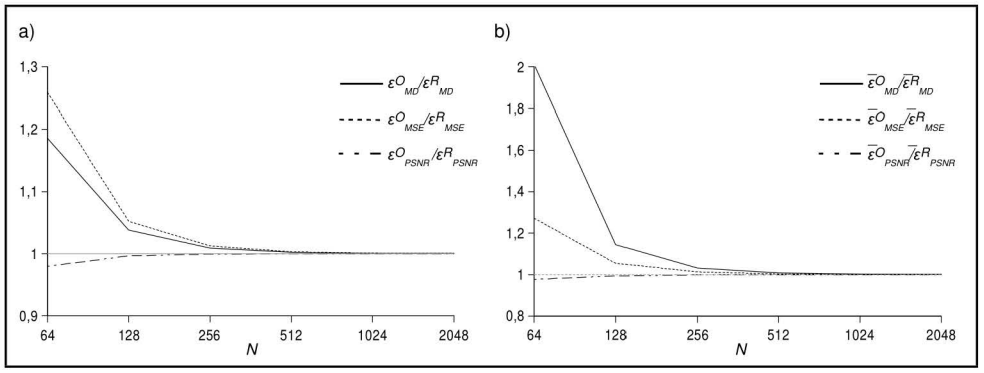
Otrzymane wyniki wskazują na to, iż uzyskane oszacownia wartości błędów rzeczywistych są dla metryk MD i MSE oszacowaniami odgórnymi, natomiast dla przypadku metryki PSNR, oszacowaniami oddolnymi. Ponadto uzyskane oszacowania są dość zgrubne jedynie dla dwóch pierwszych etapów adaptacji.

Dla ostatniego sygnału modelowego $x_4(t)$ widmo całkowego przekształcenia Fouriera przyjmuje postać

$$X_4(\omega_k) = \begin{cases} \dfrac{1}{2} & \text{dla} \quad k = 0, \\ \dfrac{1}{2\pi k}\left(sin\left(\dfrac{3}{2}\pi k\right) - sin\left(\dfrac{1}{2}\pi k\right)\right) & \text{dla} \quad k \neq 0. \end{cases}$$

Tabela 7.6: Wyniki oceny błędów względnych dla algorytmu AFFT i funkcji $x_3(t)$

N	$\bar{\epsilon}^O_{MD}$ [%]	$\bar{\epsilon}^R_{MD}$ [%]	$\bar{\epsilon}^O_{MSE} \cdot 10^9$ [%]	$\bar{\epsilon}^R_{MSE} \cdot 10^9$ [%]	$\bar{\epsilon}^O_{PSNR}$ [dB]	$\bar{\epsilon}^R_{PSNR}$ [dB]
64	37.7072	20.2137	6713.8654	4333.1778	65.7097	67.6113
128	5.2188	4.6431	276.7993	253.5653	79.5577	79.9385
256	1.1690	1.1372	15.9305	15.6022	91.9571	92.0476
512	0.2848	0.2829	0.9764	0.9714	104.0832	104.1055
1024	0.0707	0.0706	0.0607	0.0607	116.1453	116.1509
2048	0.0177	0.0177	0.0038	0.0038	128.1917	128.1931



Rysunek 7.5: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_4(t)$ i algorytmu AFFT: metryki bezwzględne (a), metryki względne (b)

Tabela 7.7: Wyniki oceny błędów bezwzględnych dla algorytmu AFFT i funkcji $x_4(t)$

N	$\epsilon^O_{MD} \cdot 10^5$	$\epsilon^R_{MD} \cdot 10^5$	$\epsilon^O_{MSE} \cdot 10^8$	$\epsilon^R_{MSE} \cdot 10^8$	$\epsilon^O_{PSNR}[dB]$	$\epsilon^R_{PSNR}[dB]$
64	472.0558	398.1128	367.8461	292.1852	48.3227	49.3228
128	100.4538	96.7514	18.4930	17.5775	61.3093	61.5298
256	24.2429	24.0228	1.1020	1.0885	73.5577	73.6112
512	6.0091	5.9955	0.0681	0.0679	85.6491	85.6624
1024	1.4991	1.4982	0.0042	0.0042	97.7028	97.7061
2048	0.3746	0.3745	0.0003	0.0003	109.7471	109.7479

Wyniki dla $x_4(t)$ zestawiono w tabelach 7.7 i 7.8 oraz przedstawiono na rysunku 7.5.

Także i w tym przypadku uzyskano wyniki analogiczne do zaprezentowanych dla sygnałów modelowych postaci $x_1(t)$, $x_2(t)$ oraz $x_3(t)$.

W dalszej części sekcji zamieszczono wyniki otrzymane za pomocą algorytmów adaptacyjnych dla całkowych kosinusowych i sinusowych przekształceń Fouriera, które obliczane były za pomocą przekształceń dyskretnych drugiego oraz czwartego rodza-

Tabela 7.8: Wyniki oceny błędów względnych dla algorytmu AFFT i funkcji $x_4(t)$

N	$\bar{\epsilon}^O_{MD}$ [%]	$\bar{\epsilon}^R_{MD}$ [%]	$\bar{\epsilon}^O_{MSE} \cdot 10^7$ [%]	$\bar{\epsilon}^R_{MSE} \cdot 10^7$ [%]	$\bar{\epsilon}^O_{PSNR}$ [dB]	$\bar{\epsilon}^R_{PSNR}$ [dB]
64	37.7072	18.7606	7527.9325	5918.5674	45.2126	46.2572
128	5.2188	4.5593	375.5827	356.0539	58.2323	58.4642
256	1.1690	1.1320	22.3367	22.0483	70.4892	70.5457
512	0.2848	0.2825	1.3793	1.3749	82.5828	82.5968
1024	0.0707	0.0706	0.0859	0.0859	94.6370	94.6405
2048	0.0177	0.0176	0.0054	0.0054	106.6815	106.6823

ju. Ponieważ sposób doboru próbek sygnału na kolejnych etapach adaptacji jest różny dla algorytmów adaptacyjnych (ADCT/ADST) (patrz algorytm 3.2), oraz szybkich algorytmów adaptacyjnych (AFCT/AFST) (patrz algorytm 5.2), to oprócz wyników uzyskanych przy pomocy algorytmów AFCT/AFST, zamieszczono również wyniki algorytmów ADCT/ADST, ale wyłącznie dla sygnałów: $x_1(t)$ (ADCT) i $x_3(t)$ (ADST). Prezentacja wyników dla jednego sygnału podyktowana jest mniejszą praktyczną użytecznością (wynikającą z większej złożoności obliczeniowej) algorytmów ADCT/ADST w stosunku do algorytmów szybkich. Jednakże przedstawienie wyników otrzymanych za pomocą algorytmów adaptacyjnych jest korzystne, ponieważ daje możliwość praktycznej weryfikacji wpływu na skuteczność szacowania błędów dodatkowych założeń heurystycznych, które wprowadzono przy konstrukcji algorytmów AFCT/AFST (patrz rozdział 5).

Szybki algorytm adaptacyjny dla dyskretnego przekształcenia kosinusowego drugiego rodzaju

Wartość całkowego kosinusowego przekształcenia Fouriera dla sygnału $x_1(t)$ i dyskretnych pulsacji ω_k^{II}, można wyznaczyć za pomocą poniższej zależności

$$X_1^C(\omega_k^{II}) = \frac{8(1-(-1)^k e^{-8})}{64+(\pi k)^2}$$

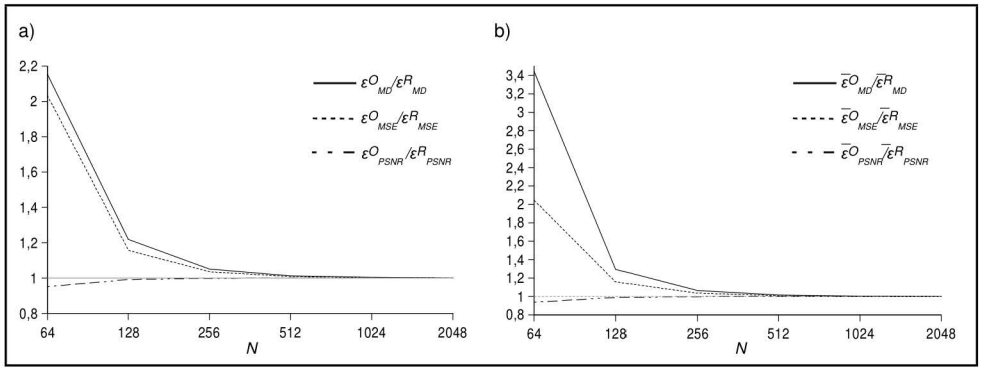
dla $k = 0, 1, ..., \infty$.

Na rysunku 7.6 oraz w tabelach 7.9 i 7.10 zamieszczono wyniki otrzymane za pomocą algorytmu adaptacyjnego dla kosinusowego przekształcenia drugiego rodzaju.

Na podstawie uzyskanych wyników widać, iż przybliżone wartości błędów są bliskie wartościom rzeczywistym, oraz co ważniejsze, uzyskano oszacowania odgórne dla metryk MD i MSE, oraz oszacowanie oddolne dla metryki PSNR.

Na rysunku 7.7 oraz w tabelach 7.11 i 7.12 zamieszczono wyniki uzyskane za pomocą algorytmu AFCT-II.

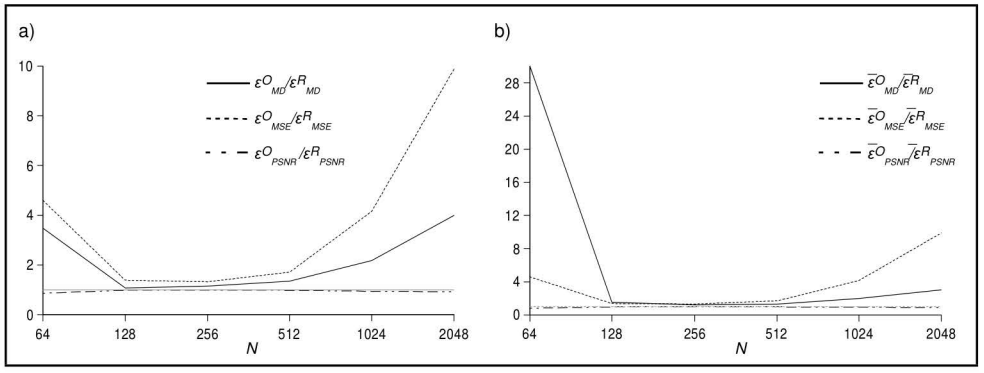
Przedstawione wyniki wskazują na to, iż w przypadku algorytmu AFCT-II uzyskano we wszystkich krokach adaptacji dla metryk MD i MSE odgórne oszacowania wartości błędów rzeczywistych i oddolne dla metryki PSNR. Jednakże przybliżone wartości błędów są bliskie wartościom rzeczywistym jedynie dla kroków $N = 128, 256$ oraz 512. W pozostałych przypadkach oszacowania są dość zgrubne i w krańcowych etapach (tj. dla $N = 64$ i 2048) stosunek przybliżonych wartości błędów do wartości rzeczywistych wynosi nawet 10.0 dla metryk bezwzględnych, oraz 30.0 dla metryk względnych.



Rysunek 7.6: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_1(t)$ i algorytmu ADCT-II: metryki bezwzględne (a), metryki względne (b)

Tabela 7.9: Wyniki oceny błędów bezwzględnych dla algorytmu ADCT-II i funkcji $x_1(t)$

N	ϵ_{MD}^{OCII}	ϵ_{MD}^{RCII}	ϵ_{MSE}^{OCII}	ϵ_{MSE}^{RCII}	ϵ_{PSNR}^{OCII} [dB]	ϵ_{PSNR}^{RCII} [dB]
64	214.5237	99.6654	15565.3803	7651.2440	60.0080	63.0980
128	26.0862	21.4069	495.8456	428.4823	74.9804	75.6160
256	5.4181	5.1527	27.0130	26.0923	87.6192	87.7702
512	1.2922	1.2760	1.6343	1.6203	99.8020	99.8392
1024	0.3193	0.3182	0.1013	0.1011	111.8781	111.8873
2048	0.0796	0.0795	0.0063	0.0063	123.9280	123.9300



Rysunek 7.7: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_1(t)$ i algorytmu AFCT-II: metryki bezwzględne (a), metryki względne (b)

Dla drugiego sygnału modelowego $x_2(t) = \sin(\pi t)$ wartości całkowego przekształcenia kosinusowego Fouriera dla pulsacji ω_k^{II} można wyznaczyć na podstawie następującej

Tabela 7.10: Wyniki oceny błędów względnych dla algorytmu ADCT-II i funkcji $x_1(t)$

N	$\bar{\epsilon}_{MD}^{OCII}$ [%]	$\bar{\epsilon}_{MD}^{RCII}$ [%]	$\bar{\epsilon}_{MSE}^{OCII} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RCII} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OCII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCII}$ [dB]
64	40.9472	11.8919	4004.2854	1959.3300	45.9044	49.0142
128	3.2995	2.5542	127.1190	109.7257	60.8918	61.5322
256	0.6547	0.6148	6.9194	6.6817	73.5342	73.6864
512	0.1547	0.1523	0.4185	0.4149	85.7179	85.7555
1024	0.0381	0.0380	0.0259	0.0259	97.7943	97.8036
2048	0.0095	0.0095	0.0016	0.0016	109.8442	109.8463

Tabela 7.11: Wyniki oceny błędów bezwzględnych dla AFCT-II i funkcji $x_1(t)$

N	$\epsilon_{MD}^{OCII} \cdot 10^5$	$\epsilon_{MD}^{RCII} \cdot 10^5$	$\epsilon_{MSE}^{OCII} \cdot 10^{10}$	$\epsilon_{MSE}^{RCII} \cdot 10^{10}$	ϵ_{PSNR}^{OCII} [dB]	ϵ_{PSNR}^{RCII} [dB]
64	215.5015	61.9879	10279.6054	2235.1647	41.8585	48.4422
128	15.7410	14.7648	151.0529	109.2130	60.1543	61.5525
256	3.8121	3.3285	7.2930	5.5187	73.3085	74.5169
512	0.8897	0.6594	0.3893	0.2279	86.0330	88.3570
1024	0.1907	0.0874	0.0187	0.0045	99.2137	105.4062
2048	0.0318	0.0080	0.0006	0.0001	113.9788	123.9300

Tabela 7.12: Wyniki oceny błędów względnych dla algorytmu AFCT-II i funkcji $x_1(t)$

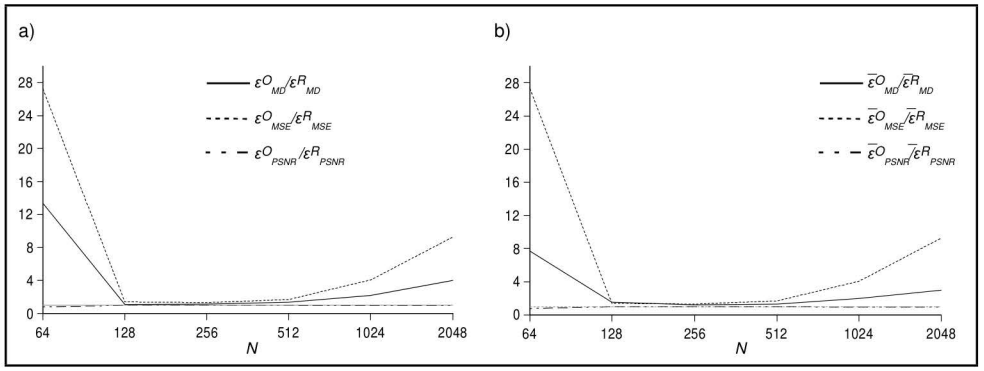
N	$\bar{\epsilon}_{MD}^{OCII}$ [%]	$\bar{\epsilon}_{MD}^{RCII}$ [%]	$\bar{\epsilon}_{MSE}^{OCII} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RCII} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OCII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCII}$ [dB]
64	1979.5711	65.8470	263527.7241	57238.0835	27.7700	34.3584
128	18.9650	12.1915	3869.1408	2796.7250	46.0694	47.4688
256	3.2013	2.6399	186.7792	141.3237	59.2243	60.4331
512	0.7026	0.5355	9.9694	5.8373	71.9490	74.2732
1024	0.1532	0.0762	0.4792	0.1152	85.1298	91.3224
2048	0.0286	0.0095	0.0160	0.0016	99.8950	109.8463

zależności

$$X_2^C(\omega_k^{II}) = \begin{cases} \dfrac{2}{(1-k^2)\pi} & \text{dla} \quad k \text{ parzystych}, \\ 0 & \text{dla} \quad k \text{ nieparzystych}. \end{cases}$$

Dla danego przypadku wyniki w postaci stosunków przybliżonych wartości błędów do wartości rzeczywistych zamieszczono na rysunku 7.8. Natomiast w tabelach 7.13 i 7.14

zamieszczono dokładne wyniki liczbowe dla bezwzględnych i względnych metryk: MD, MSE oraz PSNR.



Rysunek 7.8: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_2(t)$ i algorytmu AFCT-II: metryki bezwzględne (a), metryki względne (b)

Tabela 7.13: Wyniki oceny błędów bezwzględnych dla AFCT-II i funkcji $x_2(t)$

N	$\epsilon_{MD}^{OCII} \cdot 10^5$	$\epsilon_{MD}^{RCII} \cdot 10^5$	$\epsilon_{MSE}^{OCII} \cdot 10^{11}$	$\epsilon_{MSE}^{RCII} \cdot 10^{11}$	ϵ_{PSNR}^{OCII} [dB]	ϵ_{PSNR}^{RCII} [dB]
64	651.4705	48.7628	170879.1253	6273.1424	53.7441	68.1028
128	12.3863	11.6040	424.9840	303.8190	79.7923	81.2515
256	2.9962	2.6153	20.2739	15.3916	93.0079	94.2048
512	0.6991	0.5181	1.0831	0.6418	105.7307	108.0039
1024	0.1498	0.0687	0.0525	0.0129	118.8739	124.9712
2048	0.0250	0.0062	0.0018	0.0002	133.5108	143.1806

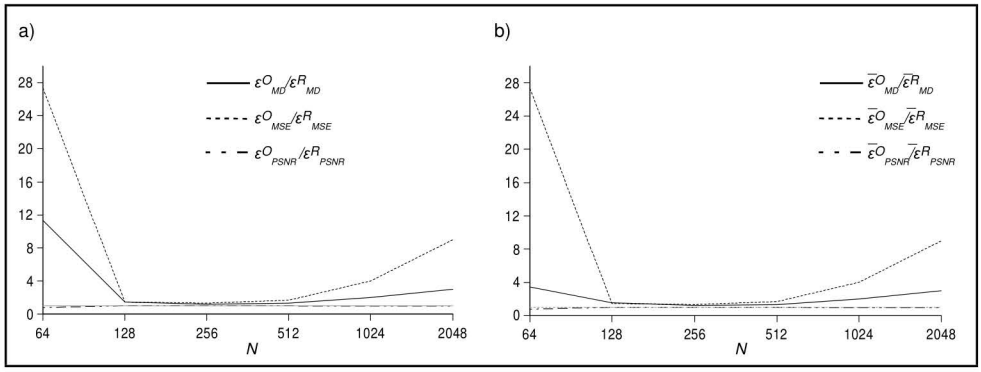
Tabela 7.14: Wyniki oceny błędów względnych dla algorytmu AFCT-II i funkcji $x_2(t)$

N	$\bar{\epsilon}_{MD}^{OCII}$ [%]	$\bar{\epsilon}_{MD}^{RCII}$ [%]	$\bar{\epsilon}_{MSE}^{OCII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{MSE}^{RCII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{PSNR}^{OCII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCII}$ [dB]
64	260.5634	33.8297	3785582.4423	138590.0717	50.2896	64.6603
128	17.0797	11.2326	9395.3942	6712.1541	76.3469	77.8090
256	2.9452	2.4403	447.9744	340.0404	89.5647	90.7623
512	0.6492	0.4956	23.9302	14.1780	102.2881	104.5615
1024	0.1418	0.0706	1.1604	0.2850	115.4314	121.5287
2048	0.0265	0.0088	0.0399	0.0043	130.0684	139.7381

Dla sygnału modelowego $x_2(t)$ uzyskano wyniki zbliżone do wyników otrzymanych dla przypadku sygnału $x_1(t)$.

Widmo całkowego kosinusowego przekształcenia Fouriera dla pulsacji ω_k^{II} oraz sygnału $x_3(t)$ można obliczyć na podstawie zależności

$$X_3^C(\omega_k^{II}) = \begin{cases} \dfrac{1}{2} & \text{dla} \quad k = 0, \\ \dfrac{4\cos\left(\frac{\pi}{2}k\right) - 2((-1)^k - 1)}{(\pi k)^2} & \text{dla} \quad k > 0. \end{cases}$$



Rysunek 7.9: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_3(t)$ i algorytmu AFCT-II: metryki bezwzględne (a), metryki względne (b)

Tabela 7.15: Wyniki oceny błędów bezwzględnych dla algorytmu AFCT-II i funkcji $x_3(t)$

N	$\epsilon_{MD}^{OCII} \cdot 10^4$	$\epsilon_{MD}^{RCII} \cdot 10^4$	$\epsilon_{MSE}^{OCII} \cdot 10^{11}$	$\epsilon_{MSE}^{RCII} \cdot 10^{11}$	ϵ_{PSNR}^{OCII} [dB]	ϵ_{PSNR}^{RCII} [dB]
64	61.2256	5.3998	135619.7623	4963.9763	52.6562	67.0211
128	1.4622	1.0133	338.8238	234.2686	78.6797	80.2823
256	0.2644	0.2201	15.6420	11.8447	92.0365	93.2441
512	0.0585	0.0447	0.8322	0.4969	104.7772	107.0167
1024	0.0128	0.0064	0.0406	0.0101	117.8962	123.9299
2048	0.0024	0.0008	0.0014	0.0002	132.4496	141.9943

Dla danego przypadku uzyskano odgórne oszacowania błędów dla metryk MD i MSE, oraz oszacowanie oddolne dla metryki PSNR. Przybliżone wartości błędów były bardzo bliskie wartościom rzeczywistym w środkowych etapach adaptacji, tj. dla $N = 128, 256$ oraz 512.

Dla sygnału modelowego $x_4(t)$ widmo całkowego kosinusowego przekształcenia Fo-

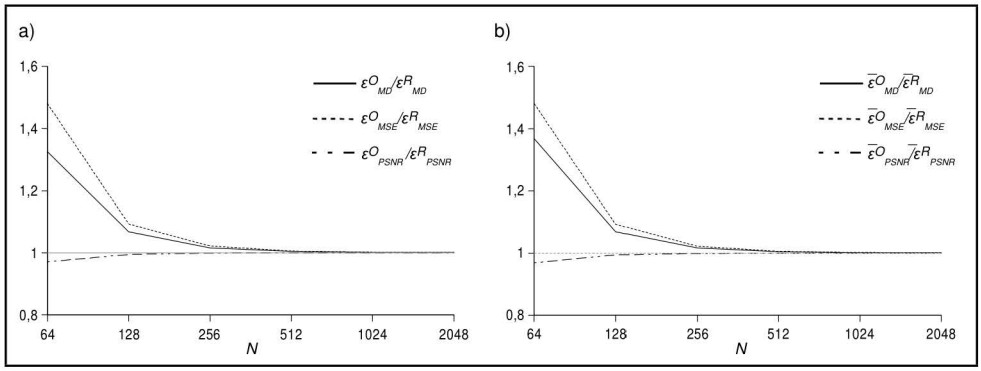
Tabela 7.16: Wyniki oceny błędów względnych dla algorytmu AFCT-II i funkcji $x_3(t)$

N	$\bar{\epsilon}_{MD}^{OCII}$ [%]	$\bar{\epsilon}_{MD}^{RCII}$ [%]	$\bar{\epsilon}_{MSE}^{OCII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{MSE}^{RCII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{PSNR}^{OCII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCII}$ [dB]
64	95.2508	27.5480	4649716.8082	170194.4413	47.3051	61.6699
128	17.0868	11.2506	11616.9986	8032.1108	73.3285	74.9311
256	2.9499	2.4444	536.3041	406.1074	86.6853	87.8930
512	0.6503	0.4964	28.5321	17.0365	99.4261	101.6656
1024	0.1420	0.0707	1.3913	0.3468	112.5451	118.5788
2048	0.0265	0.0088	0.0488	0.0054	127.0984	136.6431

uriera dla pulsacji ω_k^{II} przyjmuje następujące wartości

$$X_4^C(\omega_k^{II}) = \begin{cases} \dfrac{1}{2} & \text{dla} \quad k = 0, \\ \dfrac{\sin\left(\frac{3\pi}{4}k\right) - \sin\left(\frac{\pi}{4}k\right)}{\pi k} & \text{dla} \quad k > 0. \end{cases}$$

Wyniki uzyskane dla danego przypadku zestawiono w tabelach 7.17 i 7.18, oraz pokazano na rysunku 7.10.



Rysunek 7.10: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_4(t)$ i algorytmu AFCT-II: metryki bezwzględne (a), metryki względne (b)

W danym przypadku przybliżone wartości błędów są bliskie wartościom rzeczywistym począwszy od drugiego etapu ($N = 128$). Uzyskane oszacowania są dość zgrubne jedynie dla pierwszego kroku adaptacji. Dla wszystkich etapów oszacowania błędów są odgórne dla metryk MD i MSE, oraz oddolne dla metryki PSNR.

Szybki algorytm adaptacyjny dla dyskretnego przekształcenia kosinusowego czwartego rodzaju

Tabela 7.17: Wyniki oceny błędów bezwzględnych dla AFCT-II i funkcji $x_4(t)$

N	$\epsilon_{MD}^{OCII} \cdot 10^4$	$\epsilon_{MD}^{RCII} \cdot 10^4$	$\epsilon_{MSE}^{OCII} \cdot 10^9$	$\epsilon_{MSE}^{RCII} \cdot 10^9$	ϵ_{PSNR}^{OCII} [dB]	ϵ_{PSNR}^{RCII} [dB]
64	27.1148	20.4610	561.4300	379.2230	56.4864	58.1905
128	5.1968	4.8706	24.2234	22.1755	70.1370	70.5207
256	1.2225	1.2032	1.3934	1.3637	82.5385	82.6322
512	0.3011	0.2999	0.0853	0.0849	94.6676	94.6908
1024	0.0750	0.0749	0.0053	0.0053	106.7306	106.7364
2048	0.0187	0.0187	0.0003	0.0003	118.7772	118.7788

Tabela 7.18: Wyniki oceny błędów względnych dla algorytmu AFCT-II i funkcji $x_4(t)$

N	$\bar{\epsilon}_{MD}^{OCII}$ [%]	$\bar{\epsilon}_{MD}^{RCII}$ [%]	$\bar{\epsilon}_{MSE}^{OCII} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RCII} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OCII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCII}$ [dB]
64	13.1912	9.6420	15101.8881	10198.6144	52.1891	53.8940
128	2.4527	2.2952	651.4590	596.3758	65.8405	66.2242
256	0.5761	0.5670	37.4743	36.6749	78.2421	78.3357
512	0.1419	0.1413	2.2953	2.2830	90.3711	90.3944
1024	0.0353	0.0353	0.1427	0.1425	102.4341	102.4400
2048	0.0088	0.0088	0.0089	0.0089	114.4808	114.4824

Widmo całkowego kosinusowego przekształcenia Fouriera sygnału $x_1(t)$ dla dyskretnych wartości pulscji ω_k^{IV} przyjmuje wartości opisane poniższą zależnością

$$X_1^C(\omega_k^{IV}) = \frac{e^{-8}((-1)^k(k+\frac{1}{2})\pi)+8}{64+((k+\frac{1}{2})\pi)^2}$$

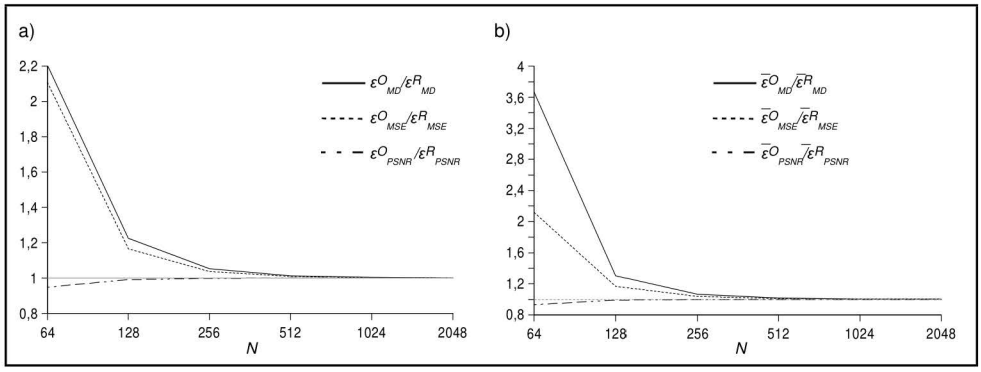
dla $k = 0, 1, ..., \infty$.

Poniżej na rysunku 7.11 oraz w tabelach 7.19 i 7.20 zamieszczono wyniki otrzymane dla algorytmu ADCT-IV i sygnału modelowego $x_1(t)$.

Otrzymane wyniki wskazują na bliskie oszacowania wartości błędu rzeczywistego dla wszystkich badanych metryk, począwszy już od drugiego etapu (od $N = 128$). Jedynie dla pierwszego etapu adaptacji otrzymano oszacowania dość zgubne. Ponadto uzyskane oszacowania błędów były zawsze odgórne dla metryk MD i MSE, oraz oddolne dla metryki PSNR.

W dalszej kolejności, tj. na rysunku 7.12 oraz w tabelach 7.21 i 7.22, zaprezentowano wyniki uzyskane przy użyciu algorytmu AFCT-IV dla sygnału modelowego $x_1(t)$.

Tutaj uzyskane wyniki wskazują na zgrubne oszacowania rzeczywistych wartości błędów w pierwszym, oraz w dwóch ostatnich etapach adaptacji. Jedynie dla etapów $N = 128, 256$ oraz 512 oszacowania można traktować jako dość bliskie wartościom rzeczywistych błędów w metrykach MD, MSE oraz PSNR. Jednakże dla wszystkich etapów uzyskane oszacowania są odgórne dla metryk MD, MSE i oddolne dla metryki PSNR.



Rysunek 7.11: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_1(t)$ i algorytmu ADCT-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.19: Ocena błędów bezwzględnych dla algorytmu ADCT-IV i funkcji $x_1(t)$

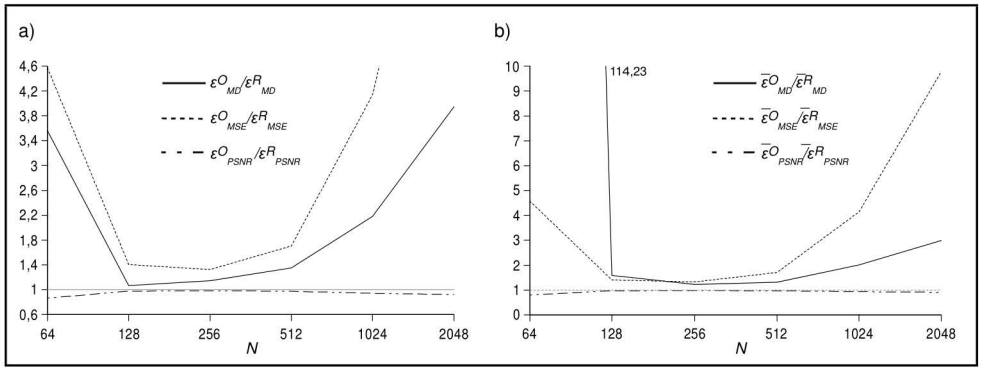
N	$\epsilon_{MD}^{OCIV} \cdot 10^1$	$\epsilon_{MD}^{RCIV} \cdot 10^1$	$\epsilon_{MSE}^{OCIV} \cdot 10^4$	$\epsilon_{MSE}^{RCIV} \cdot 10^4$	ϵ_{PSNR}^{OCIV} [dB]	ϵ_{PSNR}^{RCIV} [dB]
64	22.1811	10.0656	16255.2753	7707.2446	59.4944	62.7412
128	2.6378	2.1521	500.3459	429.2678	74.6160	75.2829
256	0.5449	0.5174	27.0740	26.1068	87.2843	87.4427
512	0.1298	0.1281	1.6353	1.6207	99.4740	99.5131
1024	0.0321	0.0319	0.1014	0.1011	111.5519	111.5615
2048	0.0080	0.0080	0.0063	0.0063	123.6022	123.6043

Tabela 7.20: Wyniki oceny błędów względnych dla algorytmu ADCT-IV i funkcji $x_1(t)$

N	$\bar{\epsilon}_{MD}^{OCIV}$ [%]	$\bar{\epsilon}_{MD}^{RCII}$ [%]	$\bar{\epsilon}_{MSE}^{OCIV} \cdot 10^9$ [%]	$\bar{\epsilon}_{MSE}^{RCII} \cdot 10^9$ [%]	$\bar{\epsilon}_{PSNR}^{OCIV}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCII}$ [dB]
64	45.6625	12.4539	52293.0790	24668.4215	44.4199	47.6888
128	3.4693	2.6627	1603.4417	1373.9486	59.5582	60.2305
256	0.6832	0.6402	86.6820	83.5595	72.2305	72.3902
512	0.1611	0.1585	5.2346	5.1874	84.4213	84.4607
1024	0.0397	0.0395	0.3244	0.3237	96.4994	96.5091
2048	0.0099	0.0099	0.0202	0.0202	108.5497	108.5518

W dalszej kolejności zaprezentowano wyniki dla algorytmu AFCT-IV i sygnałów modelowych: $x_2(t)$, $x_3(t)$ oraz $x_4(t)$. Dla sygnału $x_2(t) = \sin(\pi t)$ wartości widma całkowego przekształcenia kosinusowego Fouriera dla pulsacji ω_k^{IV} można wyznaczyć z poniższej zależności

$$X_2^C(\omega_k^{IV}) = -\frac{1}{\pi((k+\frac{1}{2})^2 - 1)}$$



Rysunek 7.12: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_1(t)$ i algorytmu AFCT-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.21: Wyniki oceny błędów bezwzględnych dla AFCT-IV i funkcji $x_1(t)$

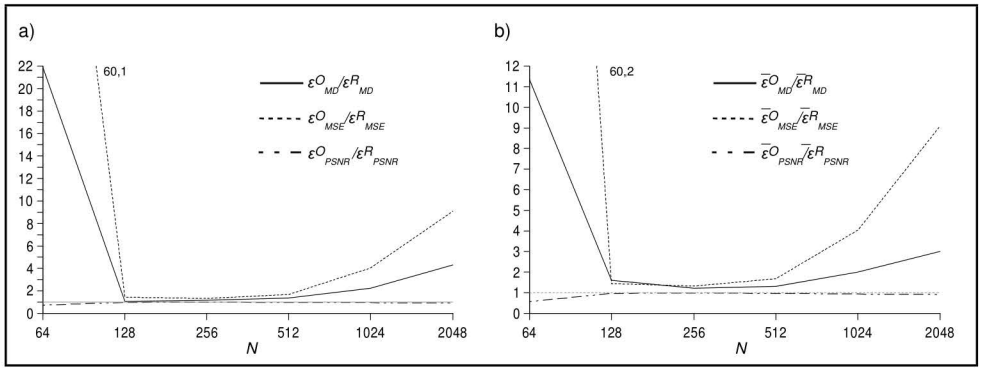
N	$\epsilon_{MD}^{OCIV} \cdot 10^4$	$\epsilon_{MD}^{RCIV} \cdot 10^4$	$\epsilon_{MSE}^{OCIV} \cdot 10^{11}$	$\epsilon_{MSE}^{RCIV} \cdot 10^{11}$	ϵ_{PSNR}^{OCIV} [dB]	ϵ_{PSNR}^{RCIV} [dB]
64	21.7608	6.1163	101748.3524	22243.9044	41.5789	48.1381
128	1.5531	1.4571	1508.4058	1074.4534	59.8355	61.2983
256	0.3762	0.3286	71.7788	54.2198	73.0526	74.2686
512	0.0878	0.0652	3.8231	2.2429	85.7865	88.1021
1024	0.0188	0.0087	0.1840	0.0444	98.9615	105.1355
2048	0.0031	0.0008	0.0062	0.0006	113.7017	123.6043

Tabela 7.22: Wyniki oceny błędów względnych dla algorytmu AFCT-IV i funkcji $x_1(t)$

N	$\bar{\epsilon}_{MD}^{OCIV}$ [%]	$\bar{\epsilon}_{MD}^{RCIV}$ [%]	$\bar{\epsilon}_{MSE}^{OCIV} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RCIV} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OCIV}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCIV}$ [dB]
64	7889.2956	69.0643	326082.0758	71195.6135	26.5210	33.0857
128	20.0325	12.6588	4829.2708	3438.9811	44.7819	46.2459
256	3.3269	2.7353	229.7676	173.5402	57.9997	59.2162
512	0.7282	0.5545	12.2373	7.1788	70.7338	73.0497
1024	0.1587	0.0788	0.5890	0.1421	83.9089	90.0831
2048	0.0296	0.0099	0.0198	0.0020	98.6493	108.5518

dla $k = 0, 1, ..., \infty$.

W danym przypadku otrzymano wyniki zbliżone do wyników uzyskanych dla sygnału modelowego $x_1(t)$. To znaczy, oszacowania błędów były zgrubne w pierwszym i w dwóch ostatnich etapach adaptacji. Dla etapów środkowych: $N = 128, 256$ oraz 512 oszacowania były bliskie rzeczywistym wartościom błędów (stosunki wartości przybliżona do rzeczywistej mniejsze od 2.0). Dla wszystkich etapów otrzymano natomiast



Rysunek 7.13: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_2(t)$ i algorytmu AFCT-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.23: Wyniki oceny błędów bezwzględnych dla AFCT-IV i funkcji $x_2(t)$

N	$\epsilon_{MD}^{OCIV} \cdot 10^4$	$\epsilon_{MD}^{RCIV} \cdot 10^4$	$\epsilon_{MSE}^{OCIV} \cdot 10^{11}$	$\epsilon_{MSE}^{RCIV} \cdot 10^{11}$	ϵ_{PSNR}^{OCIV} [dB]	ϵ_{PSNR}^{RCIV} [dB]
64	59.8621	2.7382	188380.4694	3134.5272	49.7998	67.5941
128	0.6959	0.6504	213.8907	148.1287	79.2525	80.8494
256	0.1681	0.1461	9.8954	7.4769	92.6011	93.8186
512	0.0391	0.0287	0.5259	0.3123	105.3464	107.6103
1024	0.0083	0.0037	0.0255	0.0063	118.4840	124.5543
2048	0.0014	0.0003	0.0009	0.0001	133.0843	142.6697

Tabela 7.24: Wyniki oceny błędów względnych dla algorytmu AFCT-IV i funkcji $x_2(t)$

N	$\bar{\epsilon}_{MD}^{OCIV}$ [%]	$\bar{\epsilon}_{MD}^{RCIV}$ [%]	$\bar{\epsilon}_{MSE}^{OCIV} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RCIV} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OCIV}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCIV}$ [dB]
64	403.7039	35.5744	75484.5021	1253.8160	43.7716	61.5734
128	19.9405	12.4831	85.5961	59.2517	73.2299	74.8288
256	3.2810	2.6959	3.9586	2.9908	86.5801	87.7980
512	0.7176	0.5468	0.2104	0.1249	99.3256	101.5897
1024	0.1564	0.0778	0.0102	0.0025	112.4633	118.5337
2048	0.0292	0.0097	0.0004	0.0000	127.0637	136.6491

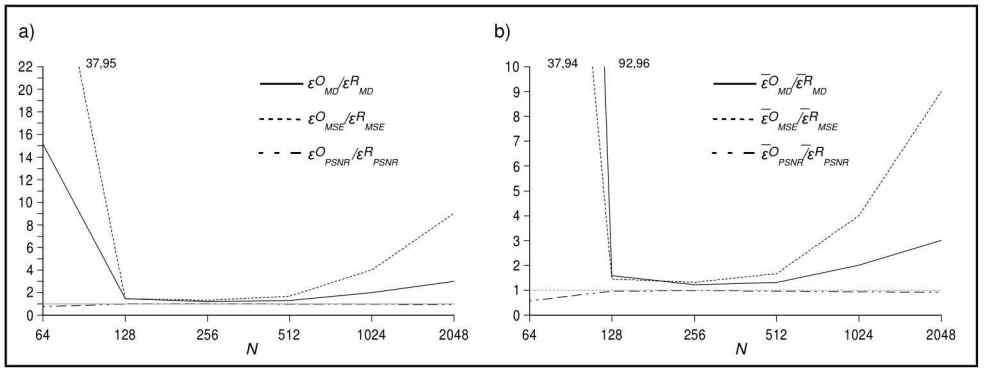
oszacowania odgórne dla metryk MD i MSE, oraz oddolne dla metryki PSNR.

Dla sygnału modelowego $x_3(t)$ całkowe przekształcenie kosinusowe Fouriera dla dyskretnych pulsacji ω_k^{IV} przyjmuje następujące wartości

$$X_3^C(\omega_k^{IV}) = \frac{2}{(\pi(k+\frac{1}{2}))^2}\left(2\cos\left(\frac{\pi}{2}(k+\frac{1}{2})\right)-1\right)$$

dla $k = 0, 1, ..., \infty$.

Na rysunku 7.14 oraz w tabelach 7.25 i 7.26 zamieszczono wyniki otrzymane dla danego sygnału modelowego i algorytmu AFCT-IV.



Rysunek 7.14: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_3(t)$ i algorytmu AFCT-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.25: Wyniki oceny błędów bezwzględnych dla algorytmu AFCT-IV i funkcji $x_3(t)$

N	$\epsilon^{OCIV}_{MD} \cdot 10^4$	$\epsilon^{RCIV}_{MD} \cdot 10^4$	$\epsilon^{OCIV}_{MSE} \cdot 10^{11}$	$\epsilon^{RCIV}_{MSE} \cdot 10^{11}$	ϵ^{OCIV}_{PSNR} [dB]	ϵ^{RCIV}_{PSNR} [dB]
64	50.0175	3.2989	141326.6631	3724.5082	49.0191	64.8096
128	0.8953	0.6131	254.2516	175.7117	76.4679	78.0723
256	0.1601	0.1330	11.7324	8.8837	89.8265	91.0344
512	0.0353	0.0270	0.6241	0.3727	102.5674	104.8070
1024	0.0077	0.0038	0.0304	0.0076	115.6865	121.7202
2048	0.0014	0.0005	0.0011	0.0001	130.2399	139.7843

Tabela 7.26: Wyniki oceny błędów względnych dla algorytmu AFCT-IV i funkcji $x_3(t)$

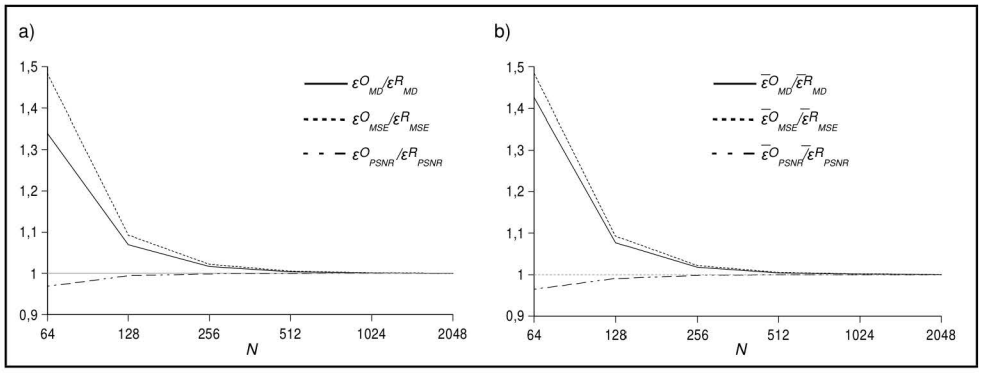
N	$\bar{\epsilon}^{OCIV}_{MD}$ [%]	$\bar{\epsilon}^{RCIV}_{MD}$ [%]	$\bar{\epsilon}^{OCIV}_{MSE} \cdot 10^{10}$ [%]	$\bar{\epsilon}^{RCIV}_{MSE} \cdot 10^{10}$ [%]	$\bar{\epsilon}^{OCIV}_{PSNR}$ [dB]	$\bar{\epsilon}^{RCIV}_{PSNR}$ [dB]
64	3656.5127	39.3324	8478843.0931	223472.1603	41.2380	57.0281
128	19.9428	12.5010	15255.4675	10542.7792	68.6863	70.2907
256	3.2856	2.7000	703.9536	533.0237	82.0449	83.2528
512	0.7187	0.5476	37.4492	22.3605	94.7859	97.0255
1024	0.1567	0.0779	1.8261	0.4552	107.9050	113.9387
2048	0.0292	0.0097	0.0640	0.0071	122.4583	132.0027

W przypadku sygnału modelowego $x_3(t)$ uzyskano wyniki zbliżone do tych, które otrzymano w poprzednich przypadkach, tj. dla sygnałów $x_1(t)$ oraz $x_2(t)$.

Dla ostatniego sygnału modelowego $x_4(t)$ widmo całkowego kosinusowego przekształcenia Fouriera dla dyskretnych pulsacji ω_k^{IV} przyjmie następującą postać

$$X_4^C(\omega_k^{IV}) = \begin{cases} \dfrac{1}{2} & \text{dla} \quad k = 0, \\ \dfrac{\sin\left(\frac{3\pi}{4}k\right) - \sin\left(\frac{\pi}{4}k\right)}{\pi k} & \text{dla} \quad k > 0. \end{cases}$$

W danym przypadku wyniki zaprezentowe zostały w tabelach 7.27 i 7.28, oraz na rysunku 7.15.



Rysunek 7.15: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_4(t)$ i algorytmu AFCT-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.27: Wyniki oceny błędów bezwzględnych dla AFCT-IV i funkcji $x_4(t)$

N	$\epsilon_{MD}^{OCIV} \cdot 10^4$	$\epsilon_{MD}^{RCIV} \cdot 10^4$	$\epsilon_{MSE}^{OCIV} \cdot 10^{10}$	$\epsilon_{MSE}^{RCIV} \cdot 10^{10}$	ϵ_{PSNR}^{OCIV} [dB]	ϵ_{PSNR}^{RCIV} [dB]
64	18.2423	13.6201	5637.4644	3799.6359	53.2341	54.9473
128	3.4611	3.2366	242.7291	222.1274	66.8935	67.2787
256	0.8125	0.7992	13.9580	13.6592	79.2965	79.3904
512	0.2000	0.1992	0.8548	0.8503	91.4258	91.4492
1024	0.0498	0.0498	0.0532	0.0531	103.4889	103.4948
2048	0.0124	0.0124	0.0033	0.0033	115.5356	115.5372

Na podstawie otrzymanych wyników można stwierdzić, iż dla sygnału modelowego $x_4(t)$ algorytm AFCT-IV dawał oszacowania bliskie wartościom rzeczywistym począwszy od drugiego etapu, i to również w etapach końcowych. Ponadto uzyskane oszacowania są odgórne dla metryk MD i MSE, oraz oddolne dla metryki PSNR.

W dalszej części sekcji przedstawione zostaną wyniki algorytmów adaptacyjnych dla sinusowego przekształcenia Fouriera, które obliczane jest za pomocą dyskretnych

Tabela 7.28: Wyniki oceny błędów względnych dla algorytmu AFCT-IV i funkcji $x_4(t)$

N	$\bar{\epsilon}_{MD}^{OCIV}$ [%]	$\bar{\epsilon}_{MD}^{RCIV}$ [%]	$\bar{\epsilon}_{MSE}^{OCIV} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RCIV} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OCIV}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCIV}$ [dB]
64	15.2593	10.7034	22845.9806	15393.3528	47.1568	48.8714
128	2.7281	2.5346	983.3774	899.8983	60.8175	61.2028
256	0.6365	0.6254	56.5478	55.3370	73.2205	73.3145
512	0.1565	0.1558	3.4632	3.4446	85.3499	85.3733
1024	0.0390	0.0389	0.2154	0.2151	97.4130	97.4189
2048	0.0097	0.0097	0.0134	0.0134	109.4597	109.4613

przekształceń sinusowych drugiego oraz czwartego rodzaju.

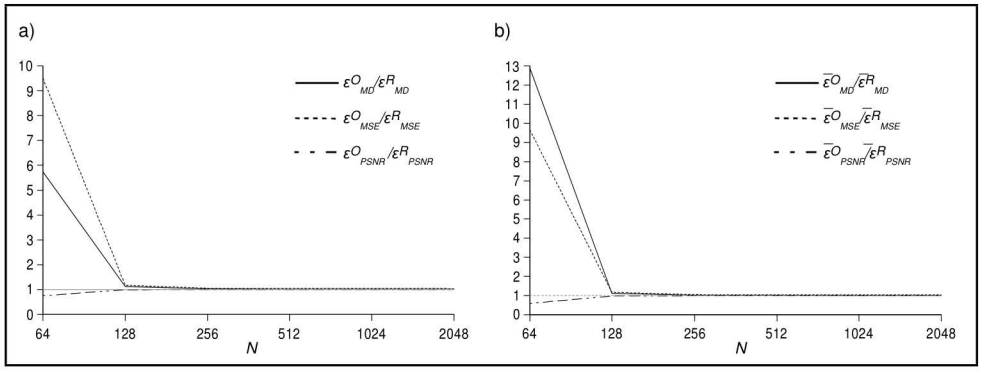
Szybki algorytm adaptacyjny dla dyskretnego przekształcenia sinusowego drugiego rodzaju

Widmo całkowego przekształcenia sinusowego dla pulsacji ω_{k+1}^{II} i sygnału modelowego $x_1(t)$ można wyznaczyć na podstawie następującej zależności

$$X_1^S(\omega_{k+1}^{II}) = \frac{\pi(k+1)(1+(-1)^k e^{-8})}{64+(\pi(k+1))^2}$$

dla $k = 0, 1, ..., \infty$.

Wyniki uzyskane dla danego sygnału modelowego przy użyciu algorytmu AFST-II zestawiono w tabelach 7.29 i 7.30, oraz pokazano na rysunku 7.16.



Rysunek 7.16: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_1(t)$ i algorytmu AFST-II: metryki bezwzględne (a), metryki względne (b)

Otrzymane wyniki wskazują na dość zgrubne oszacowanie rzeczywistych wartości błędu jedynie na pierwszym etapie adaptacji. W kolejnych etapach oszacowania są bliskie wartościom rzeczywistym i stosunki wartości przybliżonych do rzeczywistych dla metryk MD i MSE są mniejsze od 2.0, natomiast dla metryki PSNR większe od 0.95. Dla wszystkich etapów uzyskano oszacowania odgórne dla metryk MD i MSE, oraz oszacowania oddolne dla metryki PSNR.

Tabela 7.29: Wyniki oceny błędów bezwzględnych dla AFST-II i funkcji $x_1(t)$

N	$\epsilon_{MD}^{OSII} \cdot 10^4$	$\epsilon_{MD}^{RSII} \cdot 10^4$	$\epsilon_{MSE}^{OSII} \cdot 10^{10}$	$\epsilon_{MSE}^{RSII} \cdot 10^{10}$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	66.4943	11.6090	44124.9478	4641.2539	29.3872	39.1380
128	2.9826	2.6611	301.3683	257.6107	51.0205	51.6947
256	0.6697	0.6519	16.2951	15.5252	63.6854	63.8940
512	0.1634	0.1617	0.9791	0.9447	75.8967	76.0516
1024	0.0405	0.0401	0.0597	0.0571	88.0445	88.2360
2048	0.0101	0.0100	0.0036	0.0035	100.2405	100.3859

Tabela 7.30: Wyniki oceny błędów względnych dla algorytmu AFST-II i funkcji $x_1(t)$

N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^7$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^7$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	151.3728	11.7485	159513.3291	16490.3227	13.8061	23.6321
128	3.0283	2.6931	1071.1728	915.2880	35.5129	36.1888
256	0.6779	0.6597	57.8995	55.1610	48.1793	48.3880
512	0.1654	0.1636	3.4786	3.3564	60.3906	60.5456
1024	0.0410	0.0406	0.2121	0.2030	72.5385	72.7301
2048	0.0102	0.0101	0.0128	0.0124	84.7346	84.8799

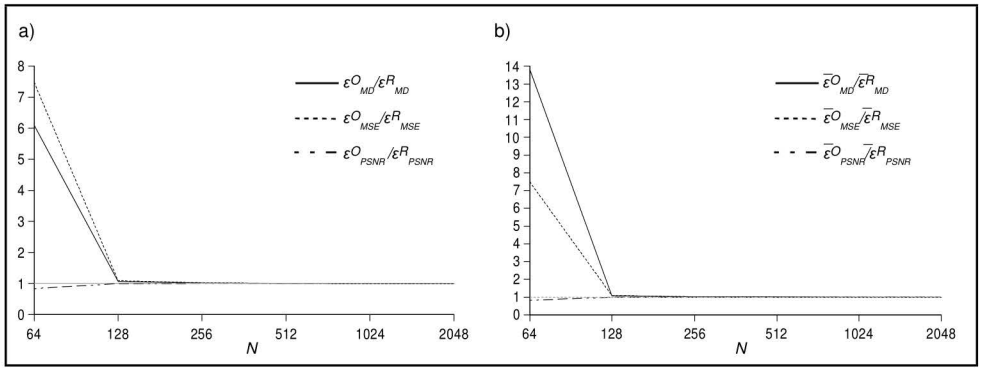
Kolejny rozpatrywamy przypadek stanowi algorytm AFST-II operujący na sygnale modelowym $x_2(t) = \cos(\pi t)$. Dla danego przypadku wartość całkowego sinusowego przekształcenia Fouriera dla pulsacji ω_{k+1}^{II} można obliczyć jako

$$X_2^S(\omega_{k+1}^{II}) = \begin{cases} 0 & \text{dla} \quad k \text{ parzystych}, \\ \dfrac{4(k+1)}{\pi k(k+2)} & \text{dla} \quad k \text{ parzystych}. \end{cases}$$

Wyniki dla sygnału modelowego $x_2(t)$ oraz algorytmu AFST-II zamieszczono w tabelach 7.31 i 7.32, oraz pokazano na wykresach w postaci stosunków wartości przybliżonych do rzeczywistych na rysunku 7.17.

W danym przypadku uzyskano wyniki zbliżone do tych, które otrzymano dla sygnału modelowego $x_1(t)$. Mianowicie oszacowania błędów w rozważanych metrykach bezwzględnych i względnych: MD, MSE oraz PSNR były dość zgrubne wyłącznie w pierwszym etapie adaptacji. Jednak wraz ze wzrostem liczby próbek sygnału asymptotycznie zbliżały się do wartości rzeczywistych. Ponadto dla wszystkich etapów uzyskano oszacowania odgórne błędów MD oraz MSE i oddolne dla metryki PSNR.

Dla sygnału modelowego $x_3(t)$ widmo całkowego przekształcenia sinusowego Fouriera dla dyskretnych wartości pulsacji ω_{k+1}^{II} przyjmuje wartości opisane poniższą zależ-



Rysunek 7.17: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_2(t)$ i algorytmu AFST-II: metryki bezwzględne (a), metryki względne (b)

Tabela 7.31: Wyniki oceny błędów bezwzględnych dla AFST-II i funkcji $x_2(t)$

N	$\epsilon_{MD}^{OSII} \cdot 10^4$	$\epsilon_{MD}^{RSII} \cdot 10^4$	$\epsilon_{MSE}^{OSII} \cdot 10^9$	$\epsilon_{MSE}^{RSII} \cdot 10^9$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	133.0512	21.8410	6135.1715	819.9519	44.6777	53.4179
128	5.5508	5.1886	52.3208	48.1071	65.3691	65.7337
256	1.3025	1.2810	3.0211	2.9639	77.7542	77.8372
512	0.3205	0.3194	0.1853	0.1851	89.8771	89.8822
1024	0.0799	0.0799	0.0116	0.0116	101.9050	101.9063
2048	0.0200	0.0200	0.0007	0.0007	113.9308	113.9309

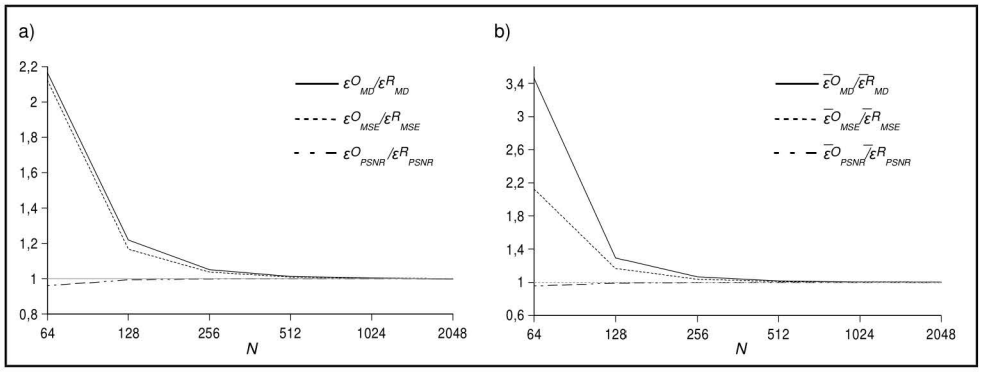
Tabela 7.32: Wyniki oceny błędów względnych dla algorytmu AFST-II i funkcji $x_2(t)$

N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^7$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^7$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	151.3184	10.9678	25188.6753	3362.4231	38.5439	47.2893
128	2.7925	2.6055	214.5590	197.2755	59.2404	59.6050
256	0.6542	0.6433	12.3887	12.1541	71.6256	71.7086
512	0.1610	0.1604	0.7599	0.7590	83.7484	83.7536
1024	0.0401	0.0401	0.0477	0.0476	95.7770	95.7776
2048	0.0100	0.0100	0.0030	0.0030	107.8022	107.8023

nością

$$X_3^S(\omega_{k+1}^{II}) = \begin{cases} \dfrac{4\sin\left(\frac{\pi}{2}(k+1)\right)}{(\pi(k+1))^2} & \text{dla} \quad k \text{ parzystych}, \\ 0 & \text{dla} \quad k \text{ nieparzystych}. \end{cases}$$

W danym przypadku jako pierwsze przedstawione zostaną wyniki uzyskane za pomocą algorytmu ADST-II. Wyniki te zestawiono w tabelach 7.33 i 7.34, oraz pokazano na wykresach z rysunku 7.18.



Rysunek 7.18: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_3(t)$ i algorytmu ADST-II: metryki bezwzględne (a), metryki względne (b)

Tabela 7.33: Wyniki oceny błędów bezwzględnych dla ADST-II i funkcji $x_3(t)$

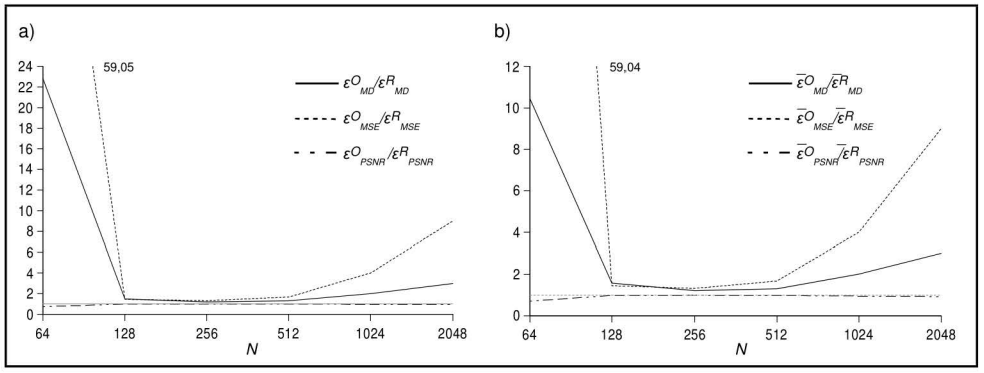
N	$\epsilon_{MD}^{OSII} \cdot 10^6$	$\epsilon_{MD}^{RSII} \cdot 10^6$	$\epsilon_{MSE}^{OSII} \cdot 10^9$	$\epsilon_{MSE}^{RSII} \cdot 10^9$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	107.9501	49.8485	2.0471	0.9641	79.0429	82.3138
128	13.0491	10.7012	0.0626	0.0537	94.1889	94.8585
256	2.7085	2.5755	0.0034	0.0033	106.8600	107.0189
512	0.6459	0.6378	0.0002	0.0002	119.0503	119.0895
1024	0.1596	0.1591	0.0000	0.0000	131.1283	131.1377
2048	0.0398	0.0398	0.0000	0.0000	143.1787	143.1795

Tabela 7.34: Wyniki oceny błędów względnych dla algorytmu ADST-II i funkcji $x_3(t)$

N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	40.9005	11.8199	12288.7806	5784.9073	71.2592	74.5322
128	3.2788	2.5374	375.6758	321.9740	86.4069	87.0770
256	0.6504	0.6107	20.3085	19.5786	99.0784	99.2374
512	0.1536	0.1512	1.2264	1.2154	111.2688	111.3079
1024	0.0379	0.0377	0.0760	0.0758	123.3468	123.3562
2048	0.0094	0.0094	0.0047	0.0047	135.3971	135.3980

Uzyskane wyniki wskazują na asymptotyczną zbieżność wartości przybliżonych błędów do ich wartosci rzeczywistych, wraz ze wzrostem liczby próbek. Ponadto uzyskane oszacowania są bliskie wartościom rzeczywistym błędów począwszy już od drugiego etapu ($N = 128$). Dla wszystkich etapów adaptacji oszacowania błędów w metrykach MD i MSE są odgórne, a w przypadku metryki PSNR oddolne.

Poniżej zestawiono wyniki otrzymane dla danego sygnału przy użyciu szybkiego algorytmu adaptacyjnego dla przekształcenia DST-II (tabela 7.35 i 7.36, rysunek 7.19).



Rysunek 7.19: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_3(t)$ i algorytmu AFST-II: metryki bezwzględne (a), metryki względne (b)

Tabela 7.35: Wyniki oceny błędów bezwzględnych dla AFST-II i funkcji $x_3(t)$

N	$\epsilon_{MD}^{OSII} \cdot 10^4$	$\epsilon_{MD}^{RSII} \cdot 10^4$	$\epsilon_{MSE}^{OSII} \cdot 10^{12}$	$\epsilon_{MSE}^{RSII} \cdot 10^{12}$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	63.1366	2.7671	1467580.91	24853.0092	50.4934	68.2014
128	0.7526	0.5093	1697.0221	1171.5653	79.8593	81.4675
256	0.1330	0.1102	78.2286	59.2261	93.2218	94.4301
512	0.0293	0.0224	4.1611	2.4845	105.9632	108.2028
1024	0.0064	0.0032	0.2029	0.0506	119.0823	125.1161
2048	0.0012	0.0004	0.0071	0.0008	133.6357	143.1795

Dla sygnału modelowego $x_3(t)$ algorytm AFST-II dał dość zgrubne oszacowania błędów dla etapu pierwszego ($N = 64$) oraz dwóch ostatnich etapów ($N = 1024, 2048$) adaptacji. W pozostałych przypadkach oszacowania były bliskie wartościom rzeczywistym, tzn. stosunki przybliżonych wartości błędów do ich wartości rzeczywistych były tutaj mniejsze od 1.7. Ponadto dla wszystkich kroków adaptacji otrzymano odgórne oszacowania błędów w metrykach MD i MSE, oraz oszacowania oddolne dla przypadku metryki PSNR.

Ostatnim sygnałem modelowym wykorzystanym podczas badań jest sygnał $x_4(t)$. Dla danego sygnału widmo całkowego przekształcenia sinusowego Fouriera dla dyskret-

Tabela 7.36: Wyniki oceny błędów względnych dla algorytmu AFST-II i funkcji $x_3(t)$

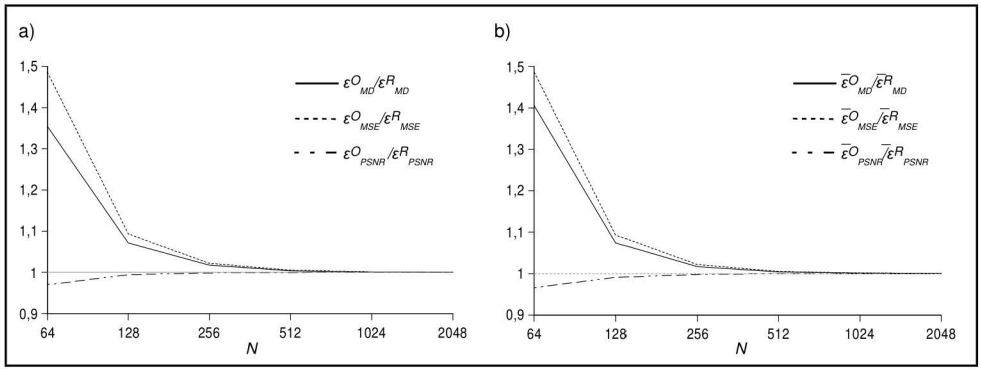
N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	684.9327	65.6134	8804276.0033	149118.8013	42.7125	60.4199
128	18.9390	12.0752	10182.3508	7029.4269	72.0777	73.6860
256	3.1711	2.6133	469.3777	355.3584	85.4402	86.6485
512	0.6955	0.5302	24.9669	14.9070	98.1816	100.4213
1024	0.1517	0.0755	1.2174	0.3034	111.3008	117.3346
2048	0.0283	0.0094	0.0427	0.0047	125.8541	135.3980

nych wartości pulsacji ω_{k+1}^{II} można wyznaczyć na podstawie następującej zależności

$$X_4^S(\omega_{k+1}^{II}) = -\frac{\cos\left(\frac{3\pi}{4}(k+1)\right) + \cos\left(\frac{\pi}{4}(k+1)\right)}{\pi(k+1)}$$

dla $k = 0, 1, ..., \infty$.

Dla sygnału $x_4(t)$ oraz algorytmu AFST-II otrzymane wyniki zestawiono w tabelach 7.37 i 7.38, oraz na rysunku 7.20.



Rysunek 7.20: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_4(t)$ i algorytmu AFST-II: metryki bezwzględne (a), metryki względne (b)

Na podstawie wyników uzyskanych dla sygnału modelowego $x_4(t)$ widać, iż algorytm AFST-II w tym przypadku dawał oszacowania bliskie wartościom rzeczywistym począwszy od drugiego etapu, a co więcej, oszacowania te asymptotycznie zbiegały do rzeczywistych wartości błędów wraz ze wzrostem liczby próbek sygnału. Ponadto podobnie, jak dla pozostałych sygnałów modelowych, otrzymane oszaczowania były odgórne dla metryk MD i MSE, oraz oddolne dla metryki PSNR.

W dalszej części sekcji zamieszczono wyniki dla ostatniego z rozważanych przekształceń dyskretnych, a mianowicie dla dyskretnego przekształcenia sinusowego czwartego rodzaju.

Tabela 7.37: Wyniki oceny błędów bezwzględnych dla AFST-II i funkcji $x_4(t)$

N	$\epsilon_{MD}^{OSII} \cdot 10^4$	$\epsilon_{MD}^{RSII} \cdot 10^4$	$\epsilon_{MSE}^{OSII} \cdot 10^{10}$	$\epsilon_{MSE}^{RSII} \cdot 10^{10}$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	20.3356	15.0182	5656.9390	3805.8699	55.5424	57.2628
128	3.8185	3.5627	243.1455	222.4409	69.2089	69.5952
256	0.8944	0.8794	13.9780	13.6777	81.6129	81.7072
512	0.2201	0.2191	0.8560	0.8514	93.7425	93.7659
1024	0.0548	0.0547	0.0532	0.0532	105.8057	105.8116
2048	0.0137	0.0137	0.0033	0.0033	117.8524	117.8539

Tabela 7.38: Wyniki oceny błędów względnych dla algorytmu AFST-II i funkcji $x_4(t)$

N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	14.5364	10.3422	22925.0472	15418.6964	49.4651	51.1868
128	2.6342	2.4534	985.0700	901.1736	63.1328	63.5192
256	0.6160	0.6056	56.6290	55.4126	75.5370	75.6312
512	0.1516	0.1509	3.4680	3.4493	87.6666	87.6900
1024	0.0377	0.0377	0.2157	0.2154	99.7297	99.7356
2048	0.0094	0.0094	0.0135	0.0135	111.7764	111.7780

Szybki algorytm adaptacyjny dla dyskretnego przekształcenia sinusowego czwartego rodzaju

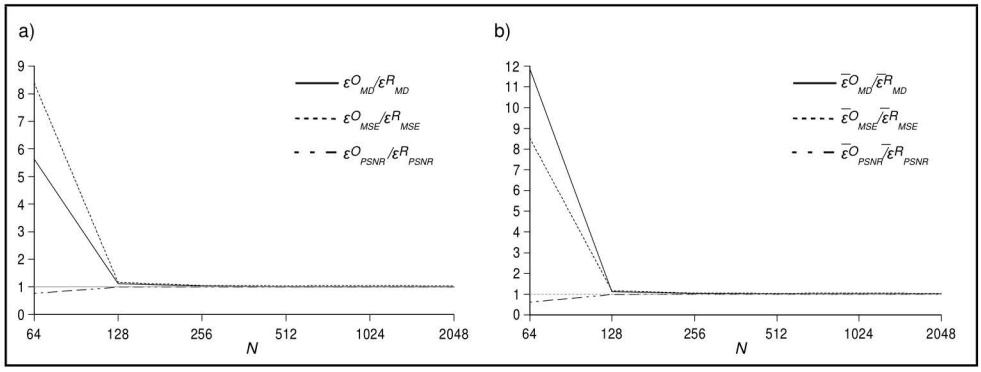
Widmo całkowego sinusowego przekształcenia Fouriera z sygnału modelowego $x_1(t)$ dla dyskretnych wartości pulsacji ω_k^{IV} można wyznaczyć na podstawie poniższej zależności

$$X_1^S(\omega_k^{IV}) = \frac{\pi(k+\frac{1}{2}) - 8e^{-8}(-1)^k}{64 + (\pi(k+\frac{1}{2})^2)}$$

dla $k = 0, 1, ..., \infty$.

Wyniki uzyskane dla sygnału $x_1(t)$ za pomocą algorytmu AFST-IV zamieszczono w tabelach 7.39, 7.40 i na rysunku 7.21.

W danym przypadku otrzymano zgrubne oszacowanie wyłącznie na pierwszym etapie adaptacji. Wraz ze wzrostem liczby próbek sygnału wejściowego przybliżona wartość błędu zbliżała się asymptotycznie do wartości rzeczywistej. Począwszy od drugiego etapu (tzn. $N = 128$ próbek) stosunki wartości przybliżonych błędów do ich wartości rzeczywistych dla matryk MD i MSE były mniejsze od 1.2, natomiast dla metryki PSNR większe niż 0.96. Ponadto dla wszystkich kroków otrzymano oszacowania odgórne błędów w metrykach MD i MSE, oraz oszacowania oddolne dla przypadku metryki PSNR.



Rysunek 7.21: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_1(t)$ i algorytmu AFST-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.39: Wyniki oceny błędów bezwzględnych dla AFST-IV i funkcji $x_1(t)$

N	$\epsilon_{MD}^{OSII} \cdot 10^4$	$\epsilon_{MD}^{RSII} \cdot 10^4$	$\epsilon_{MSE}^{OSII} \cdot 10^{10}$	$\epsilon_{MSE}^{RSII} \cdot 10^{10}$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	64.1792	11.4051	37169.8563	4426.9261	30.2385	39.4521
128	2.9282	2.6206	287.2108	246.3747	51.3377	51.9972
256	0.6595	0.6422	15.5833	14.8512	63.9881	64.1956
512	0.1610	0.1592	0.9367	0.9032	76.1975	76.3552
1024	0.0399	0.0395	0.0571	0.0546	88.3468	88.5436
2048	0.0099	0.0098	0.0034	0.0033	100.5470	100.6966

Tabela 7.40: Wyniki oceny błędów względnych dla algorytmu AFST-IV i funkcji $x_1(t)$

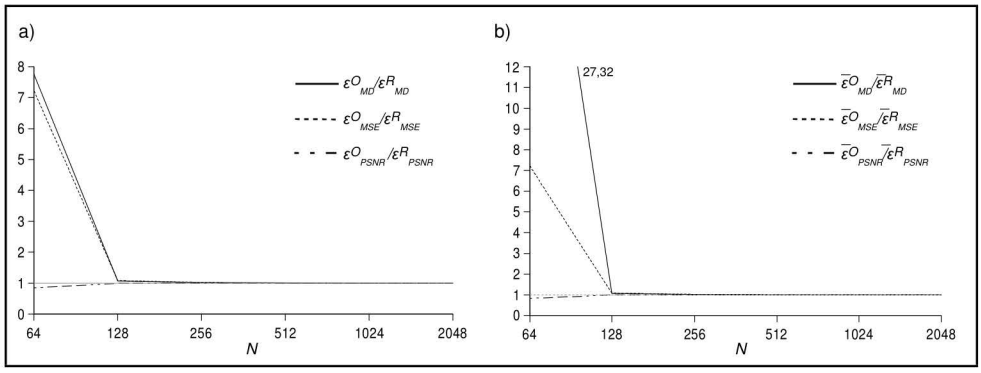
N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^7$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^7$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	134.7642	11.3599	134415.4604	15755.7247	14.6559	23.9388
128	2.9255	2.6102	1022.5873	876.8641	35.8227	36.4838
256	0.6570	0.6397	55.4651	52.8563	48.4745	48.6822
512	0.1604	0.1586	3.3338	3.2146	60.6841	60.8419
1024	0.0397	0.0394	0.2032	0.1942	72.8334	73.0302
2048	0.0099	0.0098	0.0122	0.0118	85.0336	85.1832

Dla kolejnego sygnału modelowego $x_2(t)$ widmo całkowego przekształcenia sinusowego w punktach ω_k^{IV} można wyznaczyć według zależności postaci

$$X_2^S(\omega_k^{IV}) = \frac{\pi(k+\frac{1}{2})}{(\pi(k+\frac{1}{2}))^2 - \pi^2}$$

dla $k = 0, 1, ..., \infty$.

Wyniki uzyskane dla danego przypadku zaprezentowano na wykresach z rysunku 7.22, natomiast w postaci liczbowej zamieszczono w tabelach 7.41 i 7.42.



Rysunek 7.22: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_2(t)$ i algorytmu AFST-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.41: Wyniki oceny błędów bezwzględnych dla AFST-IV i funkcji $x_2(t)$

N	$\epsilon_{MD}^{OSII} \cdot 10^4$	$\epsilon_{MD}^{RSII} \cdot 10^4$	$\epsilon_{MSE}^{OSII} \cdot 10^{10}$	$\epsilon_{MSE}^{RSII} \cdot 10^{10}$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	83.1620	10.7246	26853.4138	3720.1465	47.3493	55.9350
128	2.7242	2.5521	237.0674	219.1608	67.8916	68.2330
256	0.6406	0.6303	13.7589	13.5151	80.2547	80.3324
512	0.1577	0.1572	0.8449	0.8441	92.3724	92.3767
1024	0.0393	0.0393	0.0530	0.0530	104.4001	104.4003
2048	0.0098	0.0098	0.0033	0.0033	116.4221	116.4241

Tabela 7.42: Wyniki oceny błędów względnych dla algorytmu AFST-IV i funkcji $x_2(t)$

N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	289.6078	10.6024	108919.3620	15071.5786	41.2682	49.8590
128	2.6977	2.5230	960.5988	887.8948	61.8149	62.1570
256	0.6334	0.6232	55.7439	54.7541	74.1786	74.2565
512	0.1559	0.1554	3.4231	3.4197	86.2964	86.3008
1024	0.0388	0.0389	0.2146	0.2146	98.3240	98.3243
2048	0.0097	0.0097	0.0135	0.0135	110.3739	110.3481

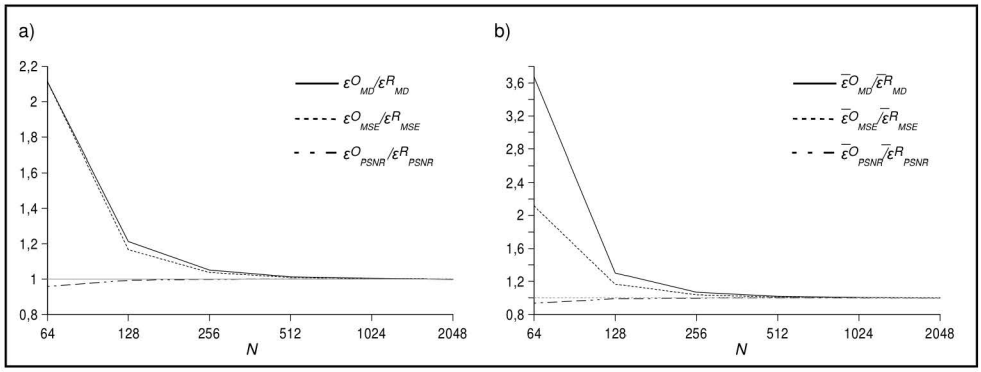
Dla sygnału $x_2(t)$ otrzymano wyniki zbliżone do wyników uzyskanych w poprzednim przypadku dla sygnału modelowego $x_1(t)$.

Jako kolejny rozpatrzony zostanie sygnał modelowy $x_3(t)$. Dla danego sygnału podano wyniki uzyskane dla algorytmów ADST-IV oraz AFST-IV. Widmo całkowego sinusowego przekształcenia Fouriera dla dyskretnych pulsacji ω_k^{IV} przyjmuje wówczas następujące wartości

$$X_3^S(\omega_k^{IV}) = \frac{4\sin\left(\frac{\pi}{2}(k+\frac{1}{2})\right) - 2(-1)^k}{(\pi(k+\frac{1}{2})^2}$$

dla $k = 0, 1, ..., \infty$.

Jako pierwsze przedstawione zostaną wyniki otrzymane dla algorytmu ADST-IV. Wyniki te zamieszczono w tabelach 7.43 i 7.44, oraz na rysunku 7.23.



Rysunek 7.23: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_3(t)$ i algorytmu ADST-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.43: Wyniki oceny błędów bezwzględnych dla ADST-IV i funkcji $x_3(t)$

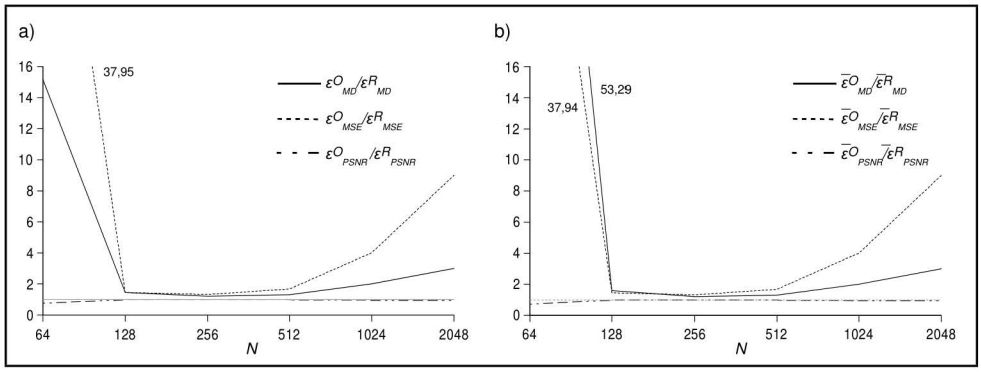
N	$\epsilon_{MD}^{OCIV} \cdot 10^5$	$\epsilon_{MD}^{RCII} \cdot 10^5$	$\epsilon_{MSE}^{OCIV} \cdot 10^{12}$	$\epsilon_{MSE}^{RCII} \cdot 10^{12}$	ϵ_{PSNR}^{OCIV} [dB]	ϵ_{PSNR}^{RCII} [dB]
64	12.6310	5.9782	3056.9176	1445.6580	75.6672	78.9196
128	1.5628	1.2897	93.8606	80.4867	90.7954	91.4630
256	0.3263	0.3108	5.0765	4.8945	103.4647	103.6232
512	0.0779	0.0770	0.3066	0.3039	115.6546	115.6937
1024	0.0193	0.0192	0.0190	0.0190	127.7325	127.7420
2048	0.0048	0.0048	0.0012	0.0012	139.7829	139.7843

Otrzymane wyniki wskazują na asymptotyczną zbieżność przybliżonych wartości błędów do wartości rzeczywistych, wraz ze wzrostem liczby próbek sygnału wejściowego. Oszacowania błędów były zgrubne wyłącznie dla pierwszego etapu. W późniejszych etapach stosunki wartości przybliżonych do rzeczywistych dla metryk MD i MSE były mniejsze od 1.3, natomiast dla metryki PSNR większe niż 0.99. Ponadto dla wszystkich etapów uzyskano odgórne oszacowania wartości błędów w metrykach MD i MSE, oraz oszacowania oddolne dla metryki PSNR.

Tabela 7.44: Wyniki oceny błędów względnych dla algorytmu ADST-IV i funkcji $x_3(t)$

N	$\bar{\epsilon}_{MD}^{OCIV}$ [%]	$\bar{\epsilon}_{MD}^{RCII}$ [%]	$\bar{\epsilon}_{MSE}^{OCIV} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{MSE}^{RCII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{PSNR}^{OCIV}$ [dB]	$\bar{\epsilon}_{PSNR}^{RCII}$ [dB]
64	45.1425	12.2854	18350.5314	8674.0128	67.8836	71.1381
128	3.4203	2.6243	563.2356	482.9240	83.0133	83.6815
256	0.6732	0.6308	30.4599	29.3673	95.6830	95.8417
512	0.1587	0.1562	1.8396	1.8231	107.8731	107.9121
1024	0.0391	0.0389	0.1140	0.1138	119.9510	119.9605
2048	0.0097	0.0097	0.0071	0.0071	132.0013	132.0027

Poniżej zamieszczono wyniki uzyskane dla algorytmu AFST-IV (patrz tabela 7.45, tabela 7.46, oraz rysunek 7.24).



Rysunek 7.24: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_3(t)$ i algorytmu AFST-IV: metryki bezwzględne (a), metryki względne (b)

Tabela 7.45: Wyniki oceny błędów bezwzględnych dla AFST-IV i funkcji $x_3(t)$

N	$\epsilon_{MD}^{OSII} \cdot 10^4$	$\epsilon_{MD}^{RSII} \cdot 10^4$	$\epsilon_{MSE}^{OSII} \cdot 10^{11}$	$\epsilon_{MSE}^{RSII} \cdot 10^{11}$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	50.0175	3.2989	141326.6631	3724.5082	49.0191	64.8096
128	0.8953	0.6131	254.2516	175.7117	76.4679	78.0723
256	0.1601	0.1330	11.7324	8.8837	89.8265	91.0344
512	0.0353	0.0270	0.6241	0.3727	102.5674	104.8070
1024	0.0077	0.0038	0.0304	0.0076	115.6865	121.7202
2048	0.0014	0.0005	0.0011	0.0001	130.2399	139.7843

Tabela 7.46: Wyniki oceny błędów względnych dla algorytmu AFST-IV i funkcji $x_3(t)$

N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^{10}$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	3656.5127	68.6172	8478843.0931	223472.1603	41.2380	57.0281
128	19.9428	12.5010	15255.4675	10542.7792	68.6863	70.2907
256	3.2856	2.7000	703.9536	533.0237	82.0449	83.2528
512	0.7187	0.5476	37.4492	22.3605	94.7859	97.0255
1024	0.1567	0.0779	1.8261	0.4552	107.9050	113.9387
2048	0.0292	0.0097	0.0640	0.0071	122.4583	132.0027

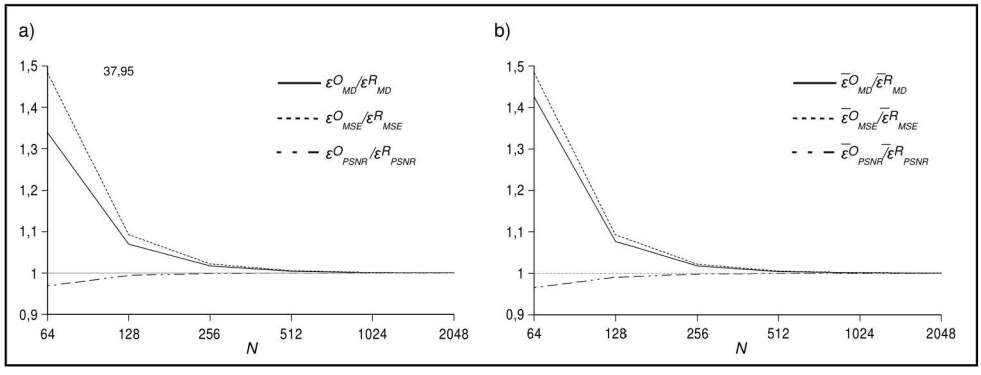
W przypadku algorytmu AFST-IV uzyskano oszacowania bliskie wartościom rzeczywistym błędów w środkowych etapach, tzn. dla $N = 128, 256$ oraz 512. W pozostałych przypadkach oszacowania były dość zgrubne, a dla pierwszego etapu stosunek wartości przybliżonej do rzeczywistej błędu wynosił nawet blisko 53.3. Dla wszystkich etapów adaptacji uzyskano oszacowania odgórne błędów MD i MSE, oraz oszacowania oddolne dla metryki PSNR.

Ostatnim sygnałem modelowym jest sygnał $x_4(t)$, dla którego widmo całkowego przekształcenia sinusowego Fouriera dla pulsacji ω_k^{IV} przyjmuje wartości następujące

$$X_4^S(\omega_k^{IV}) = \frac{\cos\left(\frac{\pi}{4}(k+\frac{1}{2})\right) - \cos\left(\frac{3\pi}{4}(k+\frac{1}{2})\right)}{\pi(k+\frac{1}{2})}$$

dla $k = 0, 1, ..., \infty$.

Wyniki dla danego przypadku zamieszczono w tabelach 7.47 i 7.48, oraz przedstawiono w postaci wykresów na rysunku 7.25. Algorytm AFST-IV dla sygnału mode-



Rysunek 7.25: Stosunki przybliżonych wartości błędów do wartości rzeczywistych dla funkcji $x_4(t)$ i algorytmu AFST-IV: metryki bezwzględne (a), metryki względne (b)

lowego $x_4(t)$ dawał oszacowania błędów bliskie wartościom rzeczywistym praktycznie dla wszystkich etapów. Dla etapu pierwszego stosunki przybliżonych wartości błędów

do wartości rzeczywistych były najmniej korzystne i wynosiły około 1.48 dla metryk MD i MSE, oraz około 0.94 dla metryki PSNR. Ponadto przybliżone wartości błędów asymptotycznie zbliżały się do wartości rzeczywistych wraz ze wzrostem liczby próbek sygnału wejściowego. Dla wszystkich etapów adaptacji uzyskano oszacowania odgórne w metrykach MD, MSE i oszacowania oddolne dla metryki PSNR.

Tabela 7.47: Wyniki oceny błędów bezwzględnych dla AFST-IV i funkcji $x_4(t)$

N	$\epsilon_{MD}^{OSII} \cdot 10^4$	$\epsilon_{MD}^{RSII} \cdot 10^4$	$\epsilon_{MSE}^{OSII} \cdot 10^{10}$	$\epsilon_{MSE}^{RSII} \cdot 10^{10}$	ϵ_{PSNR}^{OSII} [dB]	ϵ_{PSNR}^{RSII} [dB]
64	18.2423	13.6201	5637.4644	3799.6359	53.2341	54.9473
128	3.4611	3.2366	242.7291	222.1274	66.8935	67.2787
256	0.8125	0.7992	13.9580	13.6592	79.2965	79.3904
512	0.2000	0.1992	0.8548	0.8503	91.4258	91.4492
1024	0.0498	0.0498	0.0532	0.0531	103.4889	103.4948
2048	0.0124	0.0124	0.0033	0.0033	115.5356	115.5372

Tabela 7.48: Wyniki oceny błędów względnych dla algorytmu AFST-IV i funkcji $x_4(t)$

N	$\bar{\epsilon}_{MD}^{OSII}$ [%]	$\bar{\epsilon}_{MD}^{RSII}$ [%]	$\bar{\epsilon}_{MSE}^{OSII} \cdot 10^8$ [%]	$\bar{\epsilon}_{MSE}^{RSII} \cdot 10^8$ [%]	$\bar{\epsilon}_{PSNR}^{OSII}$ [dB]	$\bar{\epsilon}_{PSNR}^{RSII}$ [dB]
64	15.2593	10.7034	22845.9806	15393.3528	47.1568	48.8714
128	2.7281	2.5346	983.3774	899.8983	60.8175	61.2028
256	0.6365	0.6254	56.5478	55.3370	73.2205	73.3145
512	0.1565	0.1558	3.4632	3.4446	85.3499	85.3733
1024	0.0390	0.0389	0.2154	0.2151	97.4130	97.4189
2048	0.0097	0.0097	0.0134	0.0134	109.4597	109.4613

Analiza wpływu operacji potrzebnych do obliczania przybliżonych wartości błędów na całkowity czas realizacji obliczeń przez algorytmy adaptacyjne

W ostatniej kolejności przedstawione zostaną wyniki analizy wpływu na całkowity czas realizacji obliczeń dodatkowych operacji wymaganych przez szybkie algorytmy adaptacyjne do obliczania przybliżonych wartości błędów. W tym celu przeprowadzono szereg badań porównawczych czasów obliczania przekształceń dyskretnych poprzez algorytmy szybkie, oraz szybkie algorytmy adaptacyjne. Oczywiście w danym przypadku możliwe jest precyzyjne wyznaczenie liczby odwołań do wyrażeń służących do obliczania przybliżonych wartości błędów (patrz rozdział 6). Jednakże przez wzgląd na występowanie we wspomnianych wyrażeniach, poza trywialnymi operacjami dodawań i mnożeń, także złożonych operacji arytmetycznych, których realizacje algorytmiczne zależą od wykorzystywanych narzędzi implementacyjnych, i nie są znane autorowi, zdecydowano się na przedstawienie wyników w postaci eksperymentalnej. Z tego względu, a także z powodu przeprowadzania badań na komputerze pod kontrolą wieloprocesowego systemu operacyjnego, uzyskane rezultaty mają jedynie charakter poglądowy.

W przypadku algorytmów adaptacyjnych obliczenia prowadzono aż do ostatniego możliwego etapu. Jako maksymalne liczby próbek sygnału wejściowego przyjmowano kolejno wartości: $N = 512, 1024, 2048$ i 4096. Natomiast liczbę współczynników spektralnych, dla których szacowano błędy w bezwzględnych i względnych metrykach MD, MSE oraz PSNR, założono jako równą $M = 64$ dla przekształcenia Fouriera, oraz $M = 32$ w przypadku kosinusowych i sinusowych przekształceń drugiego i czwartego rodzaju.

Otrzymane wyniki zestawiono w tabelach 7.49-7.51 w postaci nadwyżek obliczeniowych charakterystycznych dla algorytmów adaptacyjnych, które wyrażono w procentach czasów obliczania szybkich algorytmów. Przez t_B^T oznaczono otrzymywane rezultaty, przy czym indeks T oznacza tutaj typ przekształcenia: F-dyskretne przekształcenie Fouriera, II i IV to odpowiednio dyskretne przekształcenia kosinusowe i sinusowe drugiego i czwartego rodzaju. Natomiast parametr B określa wybraną metrykę oceny błędu, tj. MD, MSE, PSNR jako metryki bezwzględne oraz $\overline{\text{MD}}$, $\overline{\text{MSE}}$ i $\overline{\text{PSNR}}$ jako metryki względne.

Zamieszczone wyniki wyznaczono jako wartości średnie z serii 100 prób przeprowadzonych dla każdego z przyjętych przypadków.

Tabela 7.49: Średnie wartości dodatkowych czasów wymaganych przez szybkie algorytmy adaptacyjne do obliczania dyskretnego przekształcenia Fouriera, przy liczbie $M = 64$ współczynników widmowych, dla których obliczano przybliżone wartości błędów

N	t^F_{MD} [%]	t^F_{MSE} [%]	t^F_{PSNR} [%]	$t^F_{\overline{MD}}$ [%]	$t^F_{\overline{MSE}}$ [%]	$t^F_{\overline{PSNR}}$ [%]
512	1.73	2.54	5.11	3.99	6.60	7.76
1024	1.62	1.96	2.32	2.90	4.57	4.87
2048	1.60	1.83	1.93	2.30	4.31	4.19
4096	1.43	1.62	1.75	1.84	3.26	3.74

Tabela 7.50: Średnie wartości dodatkowych czasów wymaganych przez szybkie algorytmy adaptacyjne do obliczania dyskretnych przekształceń kosinusowych i sinusowych drugiego rodzaju, przy liczbie $M = 32$ współczynników widmowych, dla których obliczano przybliżone wartości błędów

N	t^{II}_{MD} [%]	t^{II}_{MSE} [%]	t^{II}_{PSNR} [%]	$t^{II}_{\overline{MD}}$ [%]	$t^{II}_{\overline{MSE}}$ [%]	$t^{II}_{\overline{PSNR}}$ [%]
512	4.19	10.74	21.42	20.88	23.14	30.07
1024	3.23	9.39	11.58	12.38	14.10	17.58
2048	2.30	5.49	10.35	8.73	10.80	12.12
4096	1.75	3.60	5.00	4.59	4.84	5.69

We wszystkich przebadanych przypadkach największy wpływ dodatkowych operacji wymaganych dla oceny błędu na całkowity czas realizacji obliczeń odnotowano dla

metryki względnej PSNR. Tutaj wartości nadwyżek obliczeniowych szybkich algorytmów wahały się w granicach: od 3 do 30 procent i były największe dla dyskretnych przekształceń kosinusowych i sinusowych czwartego rodzaju dla $N = 512$. Najmniejsze nadwyżki rzędu kilku procent odnotowano dla metryki bezwzględnej MD. Natomiast w pozostałych przypadkach uzyskane wyniki leżały w granicach od 2 do 20 procent i były największe dla dyskretnych przekształceń kosinusowych i sinusowych drugiego rodzaju oraz liczby próbek $N = 512$. Takie proporcje odpowiadają spodziewanym wartościom i wynikają bezpośrednio ze złożoności obliczeniowej szybkich algorytmów dla przekształceń dyskretnych (patrz rozdział 4) oraz ze złożoności wzorów oceny błędów (patrz rozdział 6) w poszczególnych metrykach.

Tabela 7.51: Średnie wartości dodatkowych czasów wymaganych przez szybkie algorytmy adaptacyjne do obliczania dyskretnych przekształceń kosinusowych i sinusowych czwartego rodzaju, przy liczbie $M = 32$ współczynników widmowych, dla których obliczano przybliżone wartości błędów

N	t^{IV}_{MD} [%]	t^{IV}_{MSE} [%]	t^{IV}_{PSNR} [%]	$t^{IV}_{\overline{MD}}$ [%]	$t^{IV}_{\overline{MSE}}$ [%]	$t^{IV}_{\overline{PSNR}}$ [%]
512	3.74	9.99	18.24	16.05	13.32	24.68
1024	2.87	6.17	11.99	10.05	10.15	15.71
2048	2.07	5.36	7.97	6.62	6.78	9.72
4096	1.71	2.42	5.32	4.24	3.89	5.53

7.2. Wyniki badań dla przekształceń dwuwymiarowych

W przypadku przekształceń dwuwymiarowych jako sygnały modelowe wykorzystano sygnały dwóch zmiennych $x(t, s)$, które konstruowano jako iloczyny sygnałów wykorzystanych do badań nad przekształceniami jednowymiarowymi. Dla przykładu pierwszy sygnał modelowy przyjął postać funkcji: $x_1(t, s) = exp(-8t)exp(-8s)$ dla parametrów $(t, s) \in [0, 1] \times [0, 1]$. Za maksymalne liczby próbek sygnału wejściowego $x(t, s)$ przyjęto wartości $N = M = 2048$, natomiast liczba współczynników spektralnych, dla których obliczano przybliżone wartości błędów, wynosiła $N_1 \times M_1 = 32 \times 32$. Dla wszystkich badanych przekształceń, tj. dla dyskretnego dwuwymiarowego przekształcenia Fouriera oraz dyskretnych dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju, uzyskano wyniki analogiczne do tych, które otrzymano dla przypadku sygnałów jednowymiarowych.

Druga część badań dotyczyła wpływu na całkowity czas realizacji obliczeń dodatkowych operacji potrzebnych do wyznaczania przybliżonych wartości błędów przez szybkie przekształcenia adaptacyjne. Tutaj eksperymentalnie pomierzone czasy obliczania szybkich algorytmów dla różnych liczb próbek $N = M = 512, 1024, 2048$ i 4096 oraz dla $N_1 \times M_1 = 32 \times 32$, odniesiono do czasów obliczania przekształceń dyskretnych za pomocą szybkich algorytmów z przerzedzeniem w czasie. W danym przypadku wyznaczone nadwyżki obliczeniowe dla metryk bezwzględnych i względnych MD, MSE oraz PSNR były rzędu kilku procent. Należy ponadto zaznaczyć, iż szybkie algorytmy adaptacyjne kończyły swe działanie na ostatnim możliwym etapie.

7.3. Podsumowanie i wnioski

Dla dyskretnego przekształcenia Fouriera schemat doboru próbek dla szybkiego algorytmu adaptacyjnego (patrz algorytm 5.1) jest identyczny ze sposobem doboru próbek dla algorytmu adaptacyjnego ADFT (patrz algorytm 3.1). Zatem wyniki uzyskiwane w obu przypadkach będą identyczne (oczywiście z dokładnością do błędów zaokrągleń wynikających ze skończonej precyzji reprezentacji liczb w maszynach cyfrowych. Zgodnie z pracą [88] szybki algorytm FFT, a tym samym szybki algorytm adaptacyjny, jest w tym względzie lepiej uwarunkowany numerycznie, gdyż potrzebuje mniejszej liczby operacji arytmetycznych prowadzących do uzyskania żądanego wyniku). Tutaj dla wszystkich badanych sygnałów modelowych oraz metryk względnych i bezwzględnych: MD, MSE oraz PSNR, oszacowane wartości błędów numerycznego obliczania przekształcenia całkowego przez kwadraturę DFT, były bliskie ich wartościom rzeczywistym (najbardziej zgrubne dla etapów $N = 64, 128$, gdzie stosunki: wartość oszacowana do wartość rzeczywista dla metryk MD i MSE leżały w przedziale od 1.05 do 2.0, natomiast dla metryki PSNR w przedziale od 0.85 do 0.98), i co ważniejsze, we wszystkich przypadkach były oszacowaniami odgórnymi (oddolnymi dla PSNR).

Analogiczne wyniki otrzymano dla przekształcenia dwuwymiarowego i algorytmów adaptacyjnych 3.3 oraz 5.3.

W przypadku dyskretnych przekształceń kosinusowych i sinusowych drugiego oraz czwartego rodzaju schematy doboru próbek dla algorytmu adaptacyjnego (patrz algorytm 3.2) i szybkiego algorytmu adaptacyjnego (patrz algorytm 5.2) są odmienne, co dokładnie omówiono w rozdziale 5. Stąd dodatkowo zamieszczono wyniki otrzymane za pomocą algorytmów adaptacyjnych, w przypadku każdego z badanych przekształceń dyskretnych wyłącznie dla jednego wybranego sygnału modelowego (dla pozostałych sygnałów otrzymywano wyniki analogiczne). Dla algorytmu adaptacyjnego szacowane wartości błędów, dla każdego sygnału modelowego i wszystkich rozważanych metryk, były bliskie (dość zgrubne jedynie dla etapów $N = 64, 128$, a stosunki: wartość szacowana do rzeczywista zbliżone do wyników otrzymanych dla dyskretnego przekształcenia Fouriera) wartościom rzeczywistym i zawsze były oszacowaniami odgórnymi (oddolnymi dla PSNR). Jednakże w przypadku szybkich algorytmów adaptacyjnych uzyskiwano dla części badanych sygnałów modelowych oszacowania zgrubne, choć zawsze odgórne (oddolne dla PSNR), również dla etapów końcowych, tzn. dla $N \geqslant 512$. Tutaj stosunki wartości szacowanych błędów do ich wartości rzeczywistych dla metryk: MD i MSE leżały w przedziale od 1.0 do 13.5, natomiast dla metryki PSNR w przedziale od 0.85 do 1.0. Dla etapów $N = 64, 128$ oszacowania były jeszcze bardziej zgrubne i wartości szacowane były nawet ponad 100 krotnie (dla metryk MD i MSE) większe od rzeczywistych wartości błędów. Jedynie dla etapów środkowych, tj. w danym przypadku dla $N = 256$ i $N = 512$, otrzymywane oszacowania były bliskie wartościom rzeczywistym (stosunki wartości szacowanych do rzeczywistych dla metryk MD i MSE były mniejsze od 1.71, natomiast dla PSNR większe od 0.97).

Analogiczne wyniki otrzymano również dla dwuwymiarowych przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju dla algorytmu adaptacyjnego 3.4 oraz szybkiego algorytmu adaptacyjnego 5.4.

Z badań nad wpływem na całkowity czas realizacji obliczeń dodatkowych operacji związanych z szacowaniem błędów w przypadku, gdy algorytm adaptacyjny kończył działanie na ostatnim możliwym etapie wynika, iż wpływ ten może być znaczący

dopiero w chwili, gdy liczba współczynników widmowych, dla których błąd ma być szacowany, będzie zbliżała się do całkowitej liczby próbek N. Dla przykładu, przy $M = 32$ oraz $N = 512$, dla algorytmu AFCT-II i metryki względnej PSNR (wyrażenie (6.27a) charakteryzuje największa złożoność obliczeniowa) nadwyżka obliczeniowa wynosiła około 30%. Natomiast dla $N \gg M$ nadwyżka ta wynosiła już około kilku procent całkowitego czasu potrzebnego do obliczenia szybkiego algorytmu, przy liczbie próbek N. Należy tutaj zwrócić uwagę na to, iż algorytm adaptacyjny może zakończyć działanie na wcześniejszym etapie niż etap ostatni. Zatem rozważany przypadek jest przypadkiem najmniej korzystnym.

Dla przekształceń dwuwymiarowych uzyskano wyniki zbliżone do tych, przedstawionych w tabelach 7.49 - 7.51.

Na podstawie wyników uzyskanych dla sygnałów modelowych można jednoznacznie stwierdzić, iż proponowane wyrażenia oceny błędów dają oszacowania odgórne (oddolne dla metryki PSNR), które na określonych etapach są bardzo bliskie rzeczywistym wartościom błędów. Ponadto można przyjąć, iż koszt obliczeniowy związany z oceną błędów w większości przypadków nie będzie miał znaczącego wpływu na całkowity czas realizacji obliczeń.

8

Przykład zastosowania szybkich algorytmów adaptacyjnych

W przedostatnim rozdziale niniejszej monografii przedstawiony został przykład zastosowania szybkiego algorytmu adaptacyjnego dla jednowymiarowego przekształcenia Fouriera w zadaniach klasyfikacji obiektów z wykorzystaniem deskryptorów fourierowskich (FD).

Automatyczna klasyfikacja obiektów dwuwymiarowych to ważne narzędzie w takich dziedziniach jak: diagnostyka medyczna i techniczna, kryminalistyka, wojskowość, topografia i analiza zdjęć satelitarnych, astronomia, identyfikacja osób w biometrycznych systemach autoryzacji i inne [24, 62, 74, 109, 119, 135, 141, 158]. Liczne badania porównawcze, których wyniki zamieszczono między innymi w artykułach [38, 60, 91], wykazały dużą skuteczność deskryptorów fourierowskich w zadaniach klasyfikacji obiektów. Ich podstawowymi zaletami są: łatwość uzyskania opisu konturu nieczułego na afiniczne przekształcenia obiektów w postaci: skalowania, obrotów oraz przesunięć [159], a także efekt filtracji dolnoprzepustowej, która sprawia, iż wektor cech zbudowany z deskryptorów FD dobrze odwzorowuje zaszumione kształty obiektów [91]. Jako przykłady zastosowań deskryptorów fourierowskich można podać między innymi: zadania klasyfikacji mikrozwapnień [119] oraz analizę zmian nowotworowych w piersiach [109], rekonstrukcję kształtów obiektów trójwymiarowych [141], przeszukiwanie baz obrazów, kompresja sekwencji wideo [158], automatyczną klasyfikację uzębienia w oparciu o dentystyczne zdjęcia rentgenowskie [74], identyfikację militarnych pojazdów naziemnych [135], klasyfikację kształtów topograficznych [62], rozpoznawanie pisma odręcznego [24] itp. Stąd aktualnym jest zadanie udoskonalania deskryptorów fourierowskich.

W pracy [101] przedstawiono wyniki badań eksperymentalnych jednego z możliwych sposobów takiego udoskonalenia. W celu redukcji nakładu obliczeniowego zamiast szybkiego algorytmu dyskretnego przekształcenia Fouriera do obliczania deskryptorów zaproponowano użycie szybkiego algorytmu adaptacyjnego AFFT. W danym przypadku szybki algorytm adaptacyjny umożliwia automatyczny dobór takiej liczby próbek konturu, która jest wymagana do obliczenia deskryptorów z zadaną dokładnością. W ten sposób uzyskuje się redukcję obliczeń wymaganych w procesie klasyfikacji oraz zmniejszenie rozmiaru zbioru danych o konturach klasyfikowanych obiektów.

Badania przeprowadzono na ogólnie dostępnej bazie konturów ryb morskich różnych gatunków SQUID [126], do opisu których wykorzystano sygnaturę odległości od punktu centralnego. Do klasyfikacji obiektów posłużył niesparametryzowany klasyfikator najbliższego sąsiada [137], a odległości pomiędzy wektorami cech wyrażono w metryce Euklidesowej. Wyniki otrzymane dla algorytmu AFFT porównano z wynikami dla deskryptorów obliczonych bezpośrednio z definicji dyskretnego przekształcenia Fouriera oraz przy pomocy szybkiego algorytmu z przerzedzeniem w czasie typu radix-2 (patrz rozdział 4). Wyniki zestawiono w postaci porównań czasów realizacji obliczeń oraz liczby danych o konturach w funkcji trafności klasyfikacji dla kilku wybranych długości deskryptorów: 8, 16 oraz 32 współczynniki.

8.1. Deskryptory fourierowskie

Ciągły i zamknięty dwuwymiarowy kontur K można opisać za pomocą ciągłej i okresowej (z okresem T) funkcji $z(t)$ jednej zmiennej $t \in [0, T]$, przyjmującej wartości zespolone $z(t) = x(t) + iy(t)$, będące odpowiednio współrzędnymi x i y punktów wchodzących w skład konturu K. Dla potrzeb klasyfikacji do opisu konturu często wykorzystuje się tzw. *sygnaturę odległości od punktu centralnego* (patrz [159]), którą w postaci ciągłej można zapisać jako

$$r(t) = \sqrt{(x(t) - x_c)^2 + (y(t) - y_c)^2},$$

gdzie (x_c, y_c) to punkt centralny, którego współrzędne obliczamy według wzorów

$$x_c = \frac{1}{T}\int_0^T x(t)dt, \;\; y_c = \frac{1}{T}\int_0^T y(t)dt.$$

Sygnaturę tę cechuje dobra zbieżność deskryptorów fourierowskich oraz duża trafność klasyfikacji już dla niewielkich długości deskryptorów, rzędu kilkunastu współczynników (patrz [91, 159]). Ponieważ tak skonstruowania funkcja $r(t)$ jest funkcją okresową o okresie T i przyjmuje jedynie wartości rzeczywiste, to zgodnie z pracą [25] można ją w sposób jednoznaczny odtworzyć na podstawie przeliczalnego zbioru współczynników spektralnych Fouriera (patrz rozdział 2)

$$R(\omega_k) = \int_0^T r(t)e^{-i2\pi kt/T}dt,$$

gdzie $\omega_k = 2\pi k/T$ dla $k = 0, 1, ..., +\infty$. W zadaniach klasyfikacji do opisu konturu K jako wektor cech najczęściej wybiera się pewną niewielką liczbę M (np. 8, 16, 32 współczynniki) niskoczęstotliwościowych współczynników $R(\omega_k)$ dla $k = 0, 1, ..., M-1$, otrzymując tym samym deskryptor fourierowski dla konturu K.

W praktyce cyfrowego przetwarzania danych nie dysponuje się konturem w postaci ciągłej, lecz zbiorem jego dyskretnych próbek. Załóżmy zatem, że sygnatura $r(t)$ została spróbkowana równomiernie z krokiem $\Delta t = T/N$, gdzie N to liczba próbek pobieranych w dyskretnych chwilach $t_n = n\Delta t$ dla $n = 0, 1, ..., N-1$. Wówczas sygnatura odległości od punktu centralnego przyjmuje dyskretną postać, którą symbolicznie oznaczamy w następujący sposób

$$r(n) = \sqrt{(x(n) - x_c)^2 + (y(n) - y_c)^2}, \tag{8.1}$$

gdzie $r(n) \equiv r(t_n)$, $x(n) \equiv x(t_n)$ i $y(n) \equiv y(t_n)$, natomiast punkt (x_c, y_c) to punkt centralny, którego współrzędne wyznaczamy zgodnie z zależnościami

$$x_c = \frac{1}{N}\sum_{n=0}^{N-1} x(n),\ y_c = \frac{1}{N}\sum_{n=0}^{N-1} y(n).$$

W rzeczywistości zachodzi relacja $r(n) \approx r(t_n)$ dla każdego n, która jest konsekwencją dyskretyzacji współrzędnych x i y punktów konturu do wartości wynikających ze skończonej rozdzielczości poziomej i pionowej obrazu. W dalszych rozważaniach błąd kwantyzacji współrzędnych został pominięty.

Wartości współczynników widmowych stanowiących deskryptory fourierowskie dla dyskretnej funkcji $r(n)$ obliczamy korzystając z dyskretnego przekształcenia Fouriera (patrz rozdział 2)

$$R_N(k) = DFT_N\{r(n)\} = \frac{1}{N}\sum_{n=0}^{N-1} r(n)e^{-i\frac{2\pi}{N}kn}, \tag{8.2}$$

gdzie w przypadku ogólnym zachodzi związek $R_N(k) \approx R(\omega_k)$, tzn. wartości współczynników $R(\omega_k)$ są obliczane z pewnym błędem. Błąd ten spowodowany jest dyskretyzacją funkcji $r(t)$ i powstaje w wyniku nakładania widm powielonych z całkowitą wielokrotnością częstotliwości dyskretyzacji $1/\Delta t$ (tj. w wyniku aliasingu). Wartość tego błędu wyrażoną w metryce B oznaczymy przez ϵ_B^R.

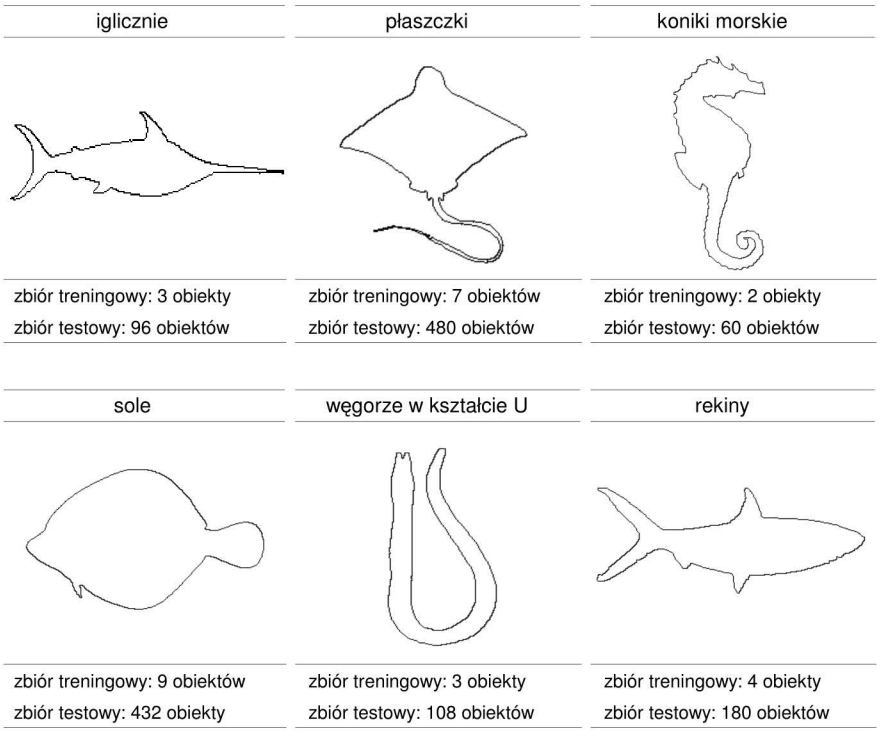
Deskryptory obliczone przy pomocy zależności (8.2) należy poddać dalszej normalizacji w celu uzyskania inwariancji na takie przekształcenia konturu K jak: zmiana skali, obrót oraz dobór punktu startowego. Inwariancję na przesunięcie konturu gwarantuje bezpośrednio konstrukcja sygnatury $r(n)$. Natomiast nieczułość na pozostałe trzy transformacje otrzymuje się, w przypadku skalowania poprzez podzielenie modułów poszczególnych współczynników przez wartość $|R(0)|$, a w przypadku obrotów i dowolności doboru punktu startowego poprzez odjęcie od argumentów współczynników $R(k)$ wartości $arg(R(1))k$ dla $k = 1, 2, ..., M-1$, tzn.

$$\bar{R}(k) = \frac{|R(k)|}{|R(0)|}e^{-i(arg(R(k))-arg(R(1))k)}, \tag{8.3}$$

gdzie przez $arg(z)$ rozumiemy argument liczby zespolonej z. W ten sposób otrzymujemy znormalizowany wektor cech postaci $\{\bar{R}(k) : k = 1, 2, ..., M-1\}$, stanowiących opis konturu K.

8.2. Badania eksperymentalne

W badaniach wykorzystano ogólnie dostępną bazę konturów ryb morskich różnych gatunków SQUID [126]. W zależności od obiektu liczba N punktów konturu wahała się w granicach od 400 do 1400. Spośród wszystkich obiektów wyłoniono sześć klas (w sumie 114 konturów), na podstawie których skonstruowano zbiory: treningowy oraz testowy. Zbiór testowy wzbogacono dodatkowo o obiekty powstałe przez przekształcenia obiektów już istniejących przy pomocy obrotów (wybrano kąty 45, 135, 225, 315), odbić w pionie i poziomie, skalowania (współczynnik skalowania dobierano w granicach od 1.5 do 2.0) oraz modyfikacji współrzędnych konturu addytywnym szumem o rozkładzie



Rysunek 8.1: Klasy obiektów wraz z przykładowymi reprezentantami oraz licznościami zbiorów: treningowego i testowego

równomiernym i wariancji 1/12. W rezultacie uzyskano zbiór testowy złożony z 1356 konturów. Rysunek 8.1 prezentuje przyjęte klasy obiektów wraz z ich przykładowymi reprezentantami.

Do opisu konturów wykorzystano sygnaturę odległości od punktu centralnego (8.1). Wektory cech skonstruowano w oparciu o deskryptory obliczone dla kilku długości $M = 8, 16$ i 32 według wzoru (8.2), które następnie poddano normalizacji zgodnie z regułą (8.3).

Ponieważ liczby N punktów konturów klasyfikowanych obiektów nie były potęgami dwóch, to do obliczania deskryptorów nie było możliwe bezpośrednie zastosowanie szybkiego algorytmu adaptacyjnego oraz szybkiego algorytmu typu radix-2. W tym celu kontury poddano interpolacji złożonymi wielomianami Lagrange'a pierwszego rzędu [7], a następnie spróbkowano je z krokami dyskretyzacji dającymi liczby próbek najbliższe, ale większe od wartości pierwotnych i będące potęgami dwójki. Do określenia klasy przynależności klasyfikowanych obiektów wykorzystano niesparametryzowany klasyfikator najbliższego sąsiada 1-NN, a odległości pomiędzy poszczególnymi deskryptorami wyrażono w metryce Euklidesowej [159].

Otrzymane wyniki podzielono na trzy kategorie związane z: deskryptorami obliczonymi dla konturów oryginalnych bezpośrednio ze wzoru (8.2) (FD), deskryptorami obliczonymi dla konturów interpolowanych (IFD) przy pomocy szybkiego algorytmu typu radix-2, oraz deskryptorami liczonymi w sposób adaptacyjny dla zadanych w me-

tryce MSE dopuszczalnych wartości błędów (AFD). Średnie czasy oraz procentową trafność klasyfikacji dla deskryptorów o długościach: 8, 16 i 32, oraz różnych wartości błędów dopuszczalnych ϵ, zestawiono w tabelach 8.1 - 8.4. Przez trafność klasyfikacji rozumiemy tutaj procentowy stosunek liczby obiektów poprawnie zaklasyfikowanych do liczności zbioru testowego

$$\text{Trafność} = \frac{\text{Liczba obiektów poprawnie zaklasyfikowanych}}{\text{Liczność zbioru testowego}} \cdot 100\%.$$

Natomiast liczba punktów AFD oznacza liczbę próbek konturu, która była potrzebna do uzyskania żądanej dokładności obliczenia deskryptorów o określonej długości M. Liczby te podano w postaci: wartości najmniejszej, średniej liczonej ze wszystkich prób oraz wartości maksymalnej.

Tabela 8.1: Średnie czasy klasyfikacji dla różnych długości deskryptorów

Długość deskryptora	Średni czas FD [ms]	Średni czas IFD [ms]	Średni czas AFD [ms]	Stosunek FD/AFD	Stosunek IFD/AFD
8	4,87	5,09	1,46	3,33	3,49
16	9,63	4,96	9,63	6,09	3,14
32	18,62	4,83	1,29	14,34	3,74

Tabela 8.2: Wyniki trafności klasyfikacji w funkcji ϵ dla przypadku $M = 8$

Dopuszczalny błąd $\epsilon \cdot 10^3$	Trafność FD [%]	Trafność IFD [%]	Trafność AFD [%]	Liczba punktów AFD		
				min.	śr.	max.
7.81250	73.75	73.75	73.6	32	64	256
3.90625	73.75	73.75	73.6	32	64	256
1.95313	73.75	73.75	74.41	32	64	256
0.97656	73.75	73.75	74.19	32	128	512
0.48828	73.75	73.75	74.78	32	128	512
0.24414	73.75	73.75	73.6	32	128	512
0.12207	73.75	73.75	73.75	32	128	512
0.06103	73.75	73.75	73.75	32	128	1024

Analiza wyników wskazuje na to, iż dla deskryptorów dłuższych niż 16-elementowe nie uzyskano poprawy trafności klasyfikacji, a szum związany z interpolacją konturów dla algorytmów IFD oraz AFD nie miał wpływu na ogólne wyniki klasyfikacji. Ponadto dla wszystkich testowanych długości deskryptorów, przy dopuszczalnym błędzie $\epsilon = 0.06103 \cdot 10^{-3}$ (co odpowiada 14 bitom), dla algorytmu AFD otrzymywano takie same trafności jak w przypadku algorytmów FD i IFD. Jednak tutaj średnia liczba

punktów konturu wynosiła: $N_1 = 128$ dla deskryptora o długości $M = 8$ oraz $N_1 = 256$ w pozostałych przypadkach. Zatem dzięki zastosowaniu algorytmu AFFT otrzymano znaczące przyspieszenie obliczeń w stosunku do podejść opartych o dyskretne przekształcenie Fouriera (około 14 krotne dla $M = 32$) oraz szybki algorytm typu radix-2 (prawie 4 krotne dla $M = 32$), co pokazuje tabela 8.1.

Tabela 8.3: Wyniki trafności klasyfikacji w funkcji ϵ dla przypadku $M = 16$

Dopuszczalny	Trafność	Trafność	Trafność	Liczba punktów AFD		
błąd $\epsilon \cdot 10^3$	FD [%]	IFD [%]	AFD [%]	min.	śr.	max.
7.81250	72.94	72.94	73.53	32	64	256
3.90625	72.94	72.94	73.45	32	64	256
1.95313	72.94	72.94	74.48	32	64	256
0.97656	72.94	72.94	74.19	32	128	512
0.48828	72.94	72.94	73.16	64	128	512
0.24414	72.94	72.94	73.23	64	128	512
0.12207	72.94	72.94	72.79	64	256	512
0.06103	72.94	72.94	72.94	128	256	1024

Tabela 8.4: Wyniki trafności klasyfikacji w funkcji ϵ dla przypadku $M = 32$

Dopuszczalny	Trafność	Trafność	Trafność	Liczba punktów AFD		
błąd $\epsilon \cdot 10^3$	FD [%]	IFD [%]	AFD [%]	min.	śr.	max.
7.81250	72.94	72.94	74.12	64	64	256
3.90625	72.94	72.94	74.26	64	64	256
1.95313	72.94	72.94	73.89	64	128	256
0.97656	72.94	72.94	73.08	64	128	512
0.48828	72.94	72.94	72.94	64	128	512
0.24414	72.94	72.94	73.01	128	128	512
0.12207	72.94	72.94	72.94	128	256	512
0.06103	72.94	72.94	72.94	128	256	1024

8.3. Podsumowanie i wnioski

Na podstawie otrzymanych wyników można jednoznacznie stwierdzić, iż użycie algorytmu AFFT do obliczania deskryptorów fourierowskich dało znaczne, bo kilku, a nawet kilkudziesięciokrotne przyspieszenie obliczeń w porównaniu z metodą bazującą na przekształceniu dyskretnym DFT, oraz przyspieszenie kilkukrotne w porównaniu z szybkim

algorytmem FFT typu radix-2. Zysk obliczeniowy wynika z faktu, iż algorytm adaptacyjny do obliczenia deskryptorów z dokładnością nie powodującą utraty trafności klasyfikacji potrzebował średnio 256 punktów konturu podczas, gdy ich liczba całkowita wahała się w granicach od 400 do 1400.

Zatem stosowanie algorytmu AFFT w zadaniach klasyfikacji może przyczynić się do przyspieszenia klasyfikacji obiektów, oraz zmniejszenia wymagań w stosunku do pojemności pamięci oraz przepustowości kanałów transmisyjnych potrzebnych do magazynowania i przesyłu danych o klasyfikowanych obiektach. Co z kolei daje nowe możliwości zastosowania deskryptorów fourierowskich w innych obszarach, gdzie obiekty opisuje się poprzez duże zbiory danych.

9

Podsumowanie

Na treść niniejszej monografii składają się opisy szybkich algorytmów adaptacyjnych dla jedno- i dwuwymiarowych dyskretnych przekształceń Fouriera oraz kosinusowych i sinusowych przekształceń drugiego i czwartego rodzaju. Algorytmy takie pozwalają na adaptacyjne obliczanie zadanego pasma widma sygnału zgodnie z kryterium *dokładność-czas realizacji obliczeń*. Celem zredukowania liczby wymaganych obliczeń rozważane algorytmy adaptacyjne bazują na szybkich algorytmach z przerzedzeniem w czasie dla obliczania dyskretnych przekształceń Fouriera i kosinusowych oraz sinusowych drugiego i czwartego rodzaju. Z kolei szybkie algorytmy z przerzedzeniem w czasie zostały opracowane na podstawie znanych szybkich algorytmów dwuetapowych jedynie poprzez zmianę kolejności przemieszania elementów wejściowych ciągów danych. Zatem charakteryzuje je taka sama, jak algorytmy dwuetapowe, złożoność obliczeniowa rzędu $\mathcal{O}(Nlog_2N)$ dla przekształceń jednowymiarowych, oraz $\mathcal{O}(N^2log_2N)$ dla przypadku dwuwymiarowego.

W celu umożliwienia oszacowania błędu numerycznego obliczania pasma widma sygnału w rozdziale 6 przedstawione zostały wyrażenia oceny błędów w popularnych metrykach powszechnie wykorzystywanych w zadaniach cyfrowego przetwarzania sygnałów, tj. w szczególności w metryce błędu średniokwadratowego (MSE), czy też szczytowego stosunku sygnału do szumu (PSNR). W rozdziale 7 zamieszczono również wyniki eksperymentalnej weryfikacji skuteczności szybkich algorytmów adaptacyjnych, wraz z wyrażeniami oceny błędów. Następnie w rozdziale 8 wskazano przykład praktycznego zastosowania szybkiego algorytmu adaptacyjnego dla przekształcenia Fouriera do zadań klasyfikacji obiektów na podstawie ich konturów w oparciu o deskryptory fourierowskie.

Kierunki dalszych badań można skoncentrować na poszukiwaniach jeszcze bardziej precyzyjnych reguł heurystycznej oceny błędów dla przypadków przekształceń kosinusowych i sinusowych drugiego i czwartego rodzaju, a także na wskazaniu kolejnych zastosowań dla proponowanych algorytmów adaptacyjnych, biorąc pod uwagę różnorodne obszary cyfrowego przetwarzania oraz analizy sygnałów.

Bibliografia

[1] Y. Abe, Y. Iiguni, *Fast Computation of the High Resolution Image Restoration by Using the Discrete Cosine Transform*, ICASSP, wol. 1, str. 745-748, 2007.

[2] E. Afshari, H. S. Bhat, A. Hajimiri, *Ultrafast analog Fourier transform using two-dimensional LC lattice*, IEEE Transactions on Circuits and Systems, przyjęto do publikacji.

[3] S. S. Agaian, O. Caglayan, *New Fast Fourier Transform with Linear Multiplicative Complexity*, IEEE Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, str. 252-256, listopad 2005.

[4] N. Ahmed, K. R. Rao, *Orthogonal Transforms For Digital Signal Processing*, Springer-Verlag, New York, 1975.

[5] N. Ahmed, T. Natarajan, K. Rao, *Discrete cosine transform* , IEEE Trans. Comput. wol. C-23, str. 90-93, styczeń 1974.

[6] K. Aizawa, H. Harashima, H. Miyakawa, *Adaptive Discrete Cosine Transform Coding With Vector Quantization For Color Images*, IEEE International Conference on ICASSP Acoustics, Speech, and Signal Processing, wol. 11, str. 985-988, kwiecień 1986.

[7] M. B. Allen III, E. L. Isaacson, *Numerical Analysis For Applied Science*, John Wiley & Sons, Inc., 1997.

[8] B. Arambepola, P. J. W. Rayner, *Multidimensional Fast-Fourier Transform Algorithm*, IEEE Electronics Letters wol. 15, nr. 13, str. 382-383, czerwiec 1979.

[9] J. Arsac, *Fourier Transforms and the Theory of Distributions*, Prentice-Hall, Inc., New Jersey, 1966.

[10] F. Beaufays, *Transform-Domain Adaptive Filters: An Analytical Approach*, IEEE Transactions on Signal Processing wol. 43, nr 2, str. 411-431, luty 1995.

[11] G. D. Bergland, *A Fast Fourier Transform Algorithm Using Base 8 Iterations*, Math. Comput. 22, str. 275-279, 1968.

[12] G. D. Bergland, *Fast Fourier Transform Hardware Implementations - An Overview*, IEEE Trans. on Audio and Electroacoustics, wol. AU-17, nr 2, str. 104-108, czerwiec 1969.

[13] L. I. Bluestein, *A Linear Filtering Approach to the Computation of the Discrete Fourier Transform*, IEEE Trans. AU-18, str. 451-455, 1970.

[14] R. Bracewell, *Przekształcenie Fouriera i jego zastosowania*, Wydawnictwa Naukowo - Techniczne, Warszawa, 1965.

[15] M. Bronstein, A. Bronstein, M. Zibulevsky, *Iterative Reconstruction in Diffraction Tomography Using Non-Uniform Fast Fourier Transform*, IEEE Proc. of International Symposium on Biomedical Imaging, str. 633-636, 2002.

[16] I. N. Brinsztejn, K. A. Siemiendiajew, *Matematyka. Poradnik encyklopedyczny*, Wydawnictwo Naukowe PWN, Warszawa, 1996.

[17] L. Brooks, *Application of The Fast Fourier Number Theoretic Transform to Radar*, IEEE, str. 137-141, 1991.

[18] G. Bruun, *z-transform DFT Filters and FFTs.*, IEEE Trans. ASSP-26, str. 56-63, 1978.

[19] A. Chahardahcherik, Y. S. Kavian, O. Strobel, R. Rejeb, *Implementing FFT Algorithms on FPGA*, IJCSNS International Journal of Computer Science and Network Security, wol. 11. nr 11, str. 148-156, listopad 2011.

[20] Joon-Hyuk Chang, *Warped Discrete Cosine Transform-Based Noisy Speech Enhancement*, IEEE Trans. on Circuits and Systems, wol. 52, nr 9, str. 535-539, wrzesień 2005.

[21] Weilong Chen, Meng Joo Er, Shiqian Wu, *Illumination Compensation and Normalization Using Logarithm and Discrete Cosine Transform*, International Conference on Control, Automation, Robotics and Vision, str. 380-385, grudzień 2004.

[22] W. H. Chen, C. H. Smith, S. C. Fralick, *A Fast Computational Algorithm For the Discrete Cosine Transform*, IEEE Trans. Commun., wol. COM-25, str. 1004-1009, wrzesień 1977.

[23] D. F. Chiper, M. N. S. Swamy, M. Omair Ahmad, T. Stouraitis, *Systolic Algorithms and a Memory-Based Design Approach for a Unified Architecture for the Computation of DCT/DST/IDCT/IDST*, IEEE Trans. on Circuits and Systems, wol. 52, nr 6, str. 1125-1137, czerwiec 2005.

[24] Yuk Ying Chung, Man To Wong, *Handwritten Character Recognition by Fourier Descriptors and Neural Network*, IEEE TENCON - Speech and Image Technologies for Computing and Telecomunications, str. 391-394, grudzień 1997.

[25] J. W. Cooley, P. A. W. Lewis, P. D. Welch, *Application of The Fast Fourier Transform to Computation of Fourier Integrals, Fourier Series, and Convolution Integrals*, IEEE Trans. on Audio and Electroacoustics, wol. AU-15, nr 2, str. 79-84, czerwiec 1967.

[26] J. W. Cooley, P. A. W. Lewis, and P. D. Welch, *The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine and Laplace Transforms*, J. Sound Vib., wol. 12, nr 3, str. 315-337, lipiec 1970.

[27] J. W. Cooley, P. A. W. Lewis, P. D. Welch, *Historical Notes on the Fast Fourier Transform*, IEEE Trans. On Audio and Electroacoustics, wol. AU-15, nr 2, str. 76-79, czerwiec 1967.

[28] J. W. Cooley, J. W. Tukey, *An Algorithm for the Machine Calculation of Complex Fourier Series*, Mathematics of Computation, wol. 19, str. 297-301, kwiecień 1965.

[29] M. J. Corinthios, *A Fast Fourier Transform for High-Speed Signal Processing*, IEEE Trans. On Computers, wol. c-20, nr 8, str. 843-846, sierpień 1971.

[30] G. Coutu, M. Dignan, *Adaptive Discrete Cosine Transform For Feedback Active Noise Control*, Conference Record of the Twenty-Ninth Asilomar Conference on Signals, Systems and Computers, wol. 1, str. 459-463, październik/listopad 1995.

[31] Zhong Cui-xiang, Han Guo-qiang, Huang Ming-he, *Some New Parallel Fast Fourier Transform Algorithms*, PDCAT Proc. 6th International Conference on Parallel and Distributed Computing, Applications and Technologies, 2005.

[32] G. Dahlquist, Å. Björck, *Metody numeryczne*, PWN, Warszawa, 1983.

[33] T. Daloze, G. Pink, H. Brunengraber, M. J. Corinthios, *Metabolic System Identification Using Simulation And Fast Fourier Transform*, IEEE Engineering In Medicine & Biology Society 10-th Conference, str. 503-504, 1988.

[34] D. Donnelly, B. Rust, *The Fast Fourier Transform For Experimentalists*, AIP and IEEE Computing in Science & Engineering, 2005.

[35] P. Duhamel, *mplementation of Split-Radix FFT Algorithms for Complex, Real, and Real-Symmetric Data*, IEEE Trans. On Acoustics. Speech and Signal Processing, wol. AsSSP-34., nr 2, str. 285-295, kwiecień 1986.

[36] M. S. Ehsani, S. E. Borujeni, *Fast Fourier Transform Speech Scrambler*, IEEE 1st International Symposium on "Intelligent Systems", str. 248-251, wrzesień 2002.

[37] T. Emoto, M. Akutagawa, U. R. Abeyratne, H. Nagashino, Y. Kinouchi, *A Comparison of Neural network and Fast Fourier Transform-based Approach for the State Analysis of Brain*, ICNN&B International Conference on Neural Networks and Brain, wol. 1, str. 94-99, 2005.

[38] N. Ezer, E. Anarim, B. Sankur, *A Comparative Study of Moment Invariants and Fourier Descriptors in Planar Shape Recognition*, Proc. 7th Mediterranean Electrotechnical Conference, kwiecień 1994.

[39] G. E. Forsyth, M. A. Malcolm, C. B. Moler, *Computer Methods For Mathematical Computations*, Prentice-Hall, Inc., New Jersey, 1977.

[40] Z. Fortuna, B. Macukow, J. Wąsowski, *Metody numeryczne*, WNT, Warszawa, 1993.

[41] D. A. Gaubatz, *FFT-Based Analog Beamforming Processor*, IEEE Proc. of Ultrasonic Symposium, str. 676-681, 1976.

[42] N. C. Geçkinli, D. Yavuz, *Discrete Fourier Transformation and Its Applications to Power Spectra Estimation*, Elsevier, Amsterdam, 1983.

[43] P. T. Gough, M. P. Hayes, *Fast Fourier techniques for SAS imagery*, IEEE OCEANSE, str. 563-568, 2005.

[44] A. M. Grigoryan, *An Algorithm for Calculation of the Discrete Cosine Transform by Paired Transform*, IEEE Trans. on Signal Processing, wol. 53, nr 1, str. 265-273, styczeń 2005.

[45] G. Gunlu, H. S. Bilge, *Driver Face Recognition by Using 3D Discrete Cosine Transform*, IEEE Intelligent Vehicles Symposium, str. 680-685, czerwiec 2007.

[46] R. M. Haralick, *A Storage Efficient Way to Implement the Discrete Cosine Transform*, IEEE Trans. Comput., wol. C-25, str. 764-765, czerwiec 1976.

[47] M. T. Heideman, *Computation of an Odd-Length DCT From a Real-Valued DFT of the Same Length*, IEEE Trans. Signal Processing, wol. 39, nr 1, str. 54-61, styczeń 1992.

[48] M. T. Heideman, D. H. Johnson, C. S. Burrus, *Gauss and the History of the Fast Fourier Transform*, IEEE ASSP Magazine, str. 14 - 21, październik 1984.

[49] A. T. Hinds, J. L. Mitchell, *A Fast and Accurate Inverse Discrete Cosine Transform*, IEEE SIPS, str. 87-92, 2005.

[50] Md. Shabiul Islam, M. Salim Beg, M. S. Bhuyan, Masuri Othman, *Design and Implementation of Discrete Cosine Transform Chip for Digital Consumer Products*, IEEE Trans. on Consumer Electronics, wol. 52, nr 3, str. 998-1003, październik 2006.

[51] ISO/IEC JTC1/SC29/WG11 N4668, *Overview of the MPEG-4 Standard*, marzec 2002.

[52] M. N. Jacymirski, *Szybkie algorytmy adaptacyjnego obliczania przekształcenia Fouriera*, Elektrotechnika Teoretyczna, wol. 46, str.40-47, 1989 (jęz. ukraiński).

[53] M. Jacymirski, *Szybkie algorytmy jednolite kosinusowych transformat drugiego i czwartego rodzaju o mnożnikach tangensowych*, AGH Automatyka, Tom 7, Zeszyt 3, str. 727-741, 2003.

[54] M. M. Jacymirski, D. Puchala, *Fast Time-Decimated Algorithms of One-Dimensional Discrete Cosine and Sine Transforms of Type II and Type IV*, Modelling and Information Technologies, nr 27, str. 137-148, Institute of Modelling Problems in Power Engineering, Ukrainian Academy of Sciences, Kiev, Ukraine, 2004.

[55] M. M. Jacymirski, D. Puchala, *Fast Time-Decimated Algorithms of Discrete Two-Dimensional Cosine And Sine Transforms of Type II And Type IV*, Modelling and Information Technologies, nr 30, str. 138-149, Institute of Modelling Problems in Power Engineering, Ukrainian Academy of Sciences, Kiev, Ukraine, 2005.

[56] M. M. Jacymirski, D. Puchala, *Szybkie adaptacyjne algorytmy jednowymiarowych przekształceń kosinusowych drugiego oraz czwartego rodzaju*, Proc. of PD FCCS'05 Conference, Selected Problems of Computer Science, str. 722-730, Exit, Warszawa 2005.

[57] Meng Joo, Weilong Chen, Shiqian Wu, *High-Speed Face Recognition Based on Discrete Cosine Transform and RBF Neural Networks*, IEEE Trans. on Neural Networks, wol. 16, nr 3, str. 679-691, maj 2005.

[58] N. N. Kachouie, J. Alirezaie, P. Fieguth, *A Hybrid Algorithm Using Discrete Cosine Transform and Gabor Filter Bank for Texture Segmentation*, IEEE Canadian Conference on Electrical and Computer Engineering, wol. 3, str. 1805-1808, maj 2004.

[59] S. Kara, A. Güven, M. Okandan, *Diagnosing Mitral and Tricuspid Stenosis with the help of Artificial Networks built on the Fast Fourier Transform Sonogram*, IEEE Proc. 1st International Conference on Neural Engineering, str. 340-434, marzec 2003.

[60] H. Kauppinen, T. Seppänen, M. Pietikäinen, *An Experimental Comparison of Autoregressive and Fourier-Based Descriptors in 2D Shape Classification*, Trans. on Pattern Analysis and Machine Intelligence, wol. 17, nr 2, str. 201 - 207, luty 1995.

[61] W. P. M. N. Keizer, *Fast Low-Sidelobe Synthesis for Large Planar Array Antennas Utilizing Successive Fast Fourier Transforms of the Array Factor*, IEEE Trans. on Antennas and Propagation, wol. 55, nr 3, str. 715-722, marzec 2007.

[62] L. Keyes, A. Winstanley, *Fourier Descriptors as a General Classification Tool for Topographic Shapes*, Proc. Irish Machine Vision and Image Processing Conference, str. 193-203, 1999.

[63] M. H. Kolekar, S. Sengupta, *Hidden Markov Model based Video Indexing with Discrete Cosine Transform as a Likelihood Function*, IEEE India Annual Conference INDICON, str. 157-159, 2004.

[64] M. Kucic, A. Low, P. Hasler, *A Programmable Continuous-Time Analog Fourier Processor Based on Floating-Gate Devices*, ISCAS 2000 - IEEE International Symposium on Circuits and Systems, str. III-351-III.354, maj 2000.

[65] I. Kuntuu, L. Lepistö, J. Rauhamaa, A. Visa, *Multiscale Fourier Descriptor for Shape Classification*, IEEE Proc. 12th International Conference on Image Analysis and Processing, str. 536-541, wrzesień 2003.

[66] B. G. Lee, *A New Algorithm For the Discrete Cosine Transform*, IEEE Trans. Acoust. Speech and Signal Processing, wol. ASSP-32, str. 1243-1245, grudzień 1983.

[67] J. Lee, N. Vijaykrishnan, M. J. Irwin, *Inverse Discrete Cosine Transform Architecture Exploiting Sparseness and Symmetry Properties*, IEEE Trans. on Circuits and Systems for Video Technology, wol. 16, nr 5, str. 655-662, maj 2006.

[68] Yong Li, Xiujuan Chen, Xuezheng Fu, Saeid Belkasim, *Multi-Level Discrete Cosine Transform for Content-Based Image Retrieval by Support Vector Machines*, ICIP, wol. 4, str. 21-24, 2007.

[69] Y. Linde, A. Buzo, R. M. Gray, *An Algorithm For Vector Quantizer Design*, IEEE Trans. Commun., wol. COM-28, str. 84-95, styczeń 1980.

[70] K.J. Ray Liu, C.T. Chiu, *A Time-Recursive DCT and DST Parallel Lattice Structure for VLSI Implementation*, 7-th MDSP Workshop, str. 10.2, wrzesień 1991.

[71] Nan Liu, Han Wang, *Recognition of Human Faces using Discrete Cosine Transform Filtered Trace Features*, Proc. IEEE International Conference on Information, Communications and Signal Processing, 2007.

[72] T. L. Lo, S. Wong, P. H. Alexander, *Fast Fourier Transform Analysis of Published ESD Waveforms and Narrowband Frequency Domain Measurement of Human ESD Events*, EOS/ESD Symposium, 1997.

[73] R. G. Lyons, *Wprowadzenie do cyfrowego przetwarzania sygnałów*, Wydawnictwa Komunikacji i Łączności, Warszawa, 2000.

[74] Mohammad Hossein Mahoor, Mohamed Abdel-Mottaleb, *Automatic Classification of Teeth in Bitewing Dental Images*, ICIP International Conference On Image Processing, str. 3475-3478, październik 2004.

[75] H. Malvar, *Lapped Transforms for Efficient Transform/Subbend Coding*, IEEE Trans. on Acoustics, Speech and Signal Processing, wol. 38, nr 6, 1990.

[76] H. S. Malvar, *Fast Computation of the Discrete Cosine Transform And the Discrete Hartley Transform*, IEEE Trans. Acoust. Speech and Signal Process., wol. ASSP-35, str. 1484-1485, wrzesień 1987.

[77] A. Materka (red.), *Elementy cyfrowego przetwarzania i analizy obrazów*, PWN, Warszawa, 1991.

[78] U. S. Mendoza-Camarena, R. de Jesús Romero-Troncoso, *VHDL Core for the Computation of the One-Dimensional Discrete Cosine Transform*, IEEE International Conference on Reconfigurable Computing and FPGA, str. 1-8, wrzesień 2006.

[79] B. J. Mohd, A. Aziz, E. E. Swartzlander Jr., *The Hazard-Free Superscalar Pipeline Fast Fourier Transform Algorithm and Architecture*, IFIP International Conference on Very Large Scale Integration, str. 194-199, 2007.

[80] J. A. Momoh, R. Button, *Design and Analysis of Aerospace DC Arcing Faults using Fast Fourier Transformation and Artificial Neural Network*, IEEE Power Engineering Society General Meeting, wol. 2, str. 788-793, lipiec 2003.

[81] M. Morháč, V. Matoušek, *Data Compression Using New Fast Adaptive Cosine-Haar transforms*, Digital Signal Processing, wol. 8, str. 63-81, 1998.

[82] M. Morháč, V. Matoušek, *An Adaptive Fast Transform Algorithm For Multi-Dimensional Data Compression*, ELSEVIER Signal Processing, wol. 43, str. 29-37, 1995.

[83] R. Muralishankar, H. N. Shankar, D. O' Shaughnessy, *A Performance Analysis of Features from Complex Cepstra of Warped DST, DCT and DHT Filters For Phoneme Recognition*, IEEE Proc. International Conference on Digital Signal Processing, str. 591-594, 2007.

[84] S. El-Din El-Nahas, A. Mottie Al Hosainy, M. M. Saeb, *Design and Implementation of Cosine Transforms Employing a CORDIC Processor*, 24-th National Radio Science Conference NRSC, wol. C16, str. 1-6, marzec 2007.

[85] R. AL Na'mneh, D. W. Pan, *Two-Step 1-D Fast Fourier Transform without Inter-Processor Communications*, IEEE Proc. 38th Southeastern Symposium on System Theory, str. 529-533, marzec 2006.

[86] M. J. Narasimha, A. M. Peterson, *On the Computation of the Discrete Cosine Transform*, IEEE Trans. Commun., wol. COM-26, str. 934-946, czerwiec 1978.

[87] A. Nukada, *FFTSS: A High Performance Fast Fourier Transform Library*, IEEE ICASSP, wol. III, str. 980-983, 2006.

[88] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer Series in Information Sciences, Springer-Verlag Berlin Heidelberg 1981.

[89] A. Ortega-Moñux, J. G. Wangüemert-Pérez, I. Molina-Fernández, *Accurate Analysis of Photonic Crystal Fibers by Means of the Fast-Fourier-Based Mode Solver*, IEEE Letters on Photonics Technology, wol. 19, nr 6, str. 414-416, marzec 2007.

[90] S. Osowski, *Sieci neuronowe w ujęciu algorytmicznym*, WNT, 1996.

[91] S. Osowski, Do Dinh Nghia, *Fourier and Wavelet Descriptors for Shape Recognition Using Neural Networks - A Comparative Study*, Pattern Recognition, wol. 35, nr 9, wrzesień 2002.

[92] P. P. Parate, N. A. Mohota, *FPGA Implementation of 2D-DCT for Image Compression*, International Journal of Science and Research, wol. 4, nr 7, str. 142-145, lipiec 2013.

[93] L. A. F. Park, M. Palaniswami, K. Ramamohanaro, *A Novel Document Ranking Method Using the Discrete Cosine Transform*, IEEE Trans. on Pattern Analysis and Machine Intelligence, wol. 27, nr 1, str. 130-135, styczeń 2005.

[94] A. Papoulis, *The Fourier Integral and Its Applications*, McGraw-Hill Book Company, Inc., New York, 1962.

[95] Ph. Philipov, V. Lazarov, Z. Zlatev, M. Ivanova, *A Parallel Architecture for Radix-2 Fast Fourier Transform*, IEEE JVA International Symposium on Modern Computing, 2006.

[96] R. J. Pogorzelski, *Improved Efficient Field Computation via Fast Fourier Transforms*, IEEE Letters on Antennas and Wireless Propagation, wol. 4, str. 27-30, 2005.

[97] M. Pryczek, P. S. Szczepaniak, *On Textual Documents Classification Using Fourier Domain Scoring*, IEEE International Conference on Web Intelligence, str. 773-777, grudzień 2006.

[98] D. Puchala, M. Yatsymirskyy, *Fast Adaptive Procedures For Two-Dimensional Cosine Transforms of Type Two and Type Four*, Information Technologies and Systems, wol. 8, nr 1, str. 29-36, Ukraina, 2005.

[99] D. Puchala, M. Yatsymirskyy, *Fast Adaptive Algorithms of One-Dimensional Discrete Sine Transforms of Type Two and Type Four*, Proc. of System Modelling Control Conference, str. 250-260, Zakopane, 2005.

[100] D. Puchala, M. M. Yatsymirskyy, *Fast Adaptive Algorithm For Two-Dimensional Fourier Transform*, Przegląd Elektrotechniczny, nr 10, str. 43-46, 2007.

[101] D. Puchala, M. M. Yatsymirskyy, *Fast Adaptive Fourier Transform for Fourier Descriptor Based Contour Classification*, Advances In Soft Computing Computer Recognition Systems 2, Proc. of CORES'07 Conference, str. 378-385, Springer, 2007.

[102] Chi-Man Pun, Hong-Min Zhu, *Textural image segmentation using discrete cosine transform*, Proceedings of the 3rd International Conference on Communications and Information Technology, str. 54-58, styczeń 2009.

[103] M. Puschel, *Cooley-Tukey FFT Like Algorithms for the DCT*, Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 2, pp. 501-504, 2003.

[104] Honggang Qi, Wen Gao, *High-Accuracy and Low-Complexity Fixed-Point Inverse Discrete Cosine Transform Based On AAN's Fast Algorithm*, IEEE ISCAS International Symposium on Circuits and Systems, str. 533-536, maj 2007.

[105] C. M. Rader, *Discrete Fourier Transforms When the Number of Data Samples Is Prime*, IEEE Proc. 56, str. 1107-1108, 1968.

[106] C. M. Rader, N. M. Brenner, *A New Principle for Fast Fourier Transform Computation*, IEEE Trans. ASSP-24, str. 264-265, 1976.

[107] A. Ralston, *Wstęp do analizy numerycznej*, Państwowe Wydawnictwo Naukowe PWN, Warszawa, 1983.

[108] G. Ramos, *Analog Computation of the Fast Fourier Transform*, Proc. of the IEEE, str. 1861-1863, listopad 1970.

[109] R. M. Rangayyan, N. M. El-Faramawy, J. E. L. Desautels, O. A. Alim, *Measures of Acutance and Shape for Classification of Breast Tumors*, IEEE Trans. on Medical Imaging, wol. 16, nr 6, str. 799-810, grudzień 1997.

[110] K. R. Rao, P. Yip, *Discrete Cosine Transform. Algorithms, Advantages Applications*, Academic Press, INC., ISBN 0-12-580203-X, 1990.

[111] Y. A. Reznik, A. T. Hinds,N. Rjavec, *Low Complexity Fixed-Point Approximation of Inverse Discrete Cosine Transform*, IEEE ICASSP, wol. 1, str. 1109-1112, 2007.

[112] M. Rosenblatt, *Procesy stochastyczne*, Państwowe Wydawnictwa Naukowe PWN, Warszawa, 1967.

[113] L. Rutkowski, *Filtry adaptacyjne i adaptacyjne przetwarzanie sygnałów: teoria i zastosowania*, WNT, Warszawa, 1994.

[114] L. Rutkowski, *Metody i techniki sztucznej inteligencji*, PWN, Warszawa, 2005.

[115] A. S. Samra, S. El T. Gad Allah, R. M. Ibrahim, *Face Recognition Using Wavelet Transform, Fast Fourier Transform and Discrete Cosine Transform*, Proc. of the 46th IEEE International Midwest Symposium on Circuits and Systems, wol. 1, str. 272-275, 2003.

[116] A. K. Sao, B. Yegnanarayana, *On the use of phase of the Fourier transform for face recognition under variations in illumination*, Signal, Image and Video Processing, wol. 4, nr 3, str. 353-358, wrzesień 2010.

[117] M. Savvides, J. Heo, R. Abiantum, C. Xie, B. V. K. V. Kumar, *Class Dependent Kernel Discrete Cosine Transform Features for Enhanced Holistic Face Recognition in FRGC-II*, ICASSP, wol. 2, str. 185-188, 2006.

[118] K. Sayood, *Kompresja danych - wprowadzenie*, Wydawnictwo RM, Warszawa 2002.

[119] Liang Shen, R. M. Rangayyan, J. E. L. Desautels, *Application of Shape Analysis to Mammographic Calcifications*, IEEE Trans. on Medical Imaging, wol. 13, nr 2, str. 263-274, czerwiec 1994.

[120] T. Sikora, B. Makai, *Shape-Adaptive DCT For Generic Coding of Video*, IEEE Trans. on Circuits and Systems for Video Technology, wol. 5, nr 1, str. 59-62, luty 1995.

[121] R. C. Singleton, *An Algorithm for Computing the Mixed Radix Fast Fourier Transform*, IEEE Trans. AU-17, str. 93-103, 1969.

[122] R. C. Singleton, *On Computing the Fast Fourier Transform*, Commun. ACM, wol. 1O, nr 1O, str. 647-654, wrzesień 1967.

[123] A. C. Quinto Siravenha, E. Golalves Pelaes, *The use of Discrete Cosine Transform for satellite images segmentation and comparison to statistical metrics*, Anais XV Simpósio Brasileiro de Sensoriamento Remoto - SBSR, str. 7271-7278, maj 2011.

[124] W. Skarbek (red.), MULTIMEDIA. *Algorytmy i standardy kompresji*, PLJ, Warszawa 1998.

[125] H. V. Sorensen, C. A. Katz, C. S. Burrus, *Efficient FFT algorithms for DSP processors using tensor product decomposition*, ICASSP International Conference on Acoustics, Speech, and Signal Processing, wol. 3, str. 1507 - 1510, kwiecień 1990.

[126] http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html

[127] Standard ISO/IEC 10918, *Digital Compression and Coding of Continuous-tone Still Images.*

[128] Standard ISO 11172, *Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 MB/s - MPEG-1.*

[129] Standard ISO 13818, *Generic Coding of Moving Pictures and Associated Audio Information MPEG-2.*

[130] Standard H.261, *Video Codec for Audiovisual Services at 64 kbit/s*, ITU-T.

[131] Standard H.263, *Video Coding for Low Bitrate Communication*, ITU-T.

[132] R. Stasidski, J. Konrad, *A New Class of Fast Shape-Adaptive Orthogonal Transforms And Their Application To Region-Based Image Compression*, IEEE Trans. on Circuits and Systems for Video Technology, wol. 9, nr 1, str. 16-34, luty 1999.

[133] E. M. Stein, R. Shakarchi, *Fourier Analysis. An Introduction*, Princeton University Press, New Jersey, 2003.

[134] N. Suehiro, M. Hatori, *Fast Algorithms For the DFT And Other Sinusoidal Transforms*, IEEE Trans. Acoust. Speech and Signal Processing, wol. ASSP-34, str. 642-644, czerwiec 1986.

[135] Sun-Gu Sun, Junsung Park, Yong Woon Park, *Identification of Military Ground Vehicles by Feature Information Fusion in FLIR Images*, Proc. 3rd International Symposium on Image and Signal Processing and Analysis, str. 871-876, 2003.

[136] J. Szabatin, *Podstawy teorii sygnałów*, Wydawnictwa Komunikacji i Łączności, Warszawa, 2000.

[137] P. Szczepaniak, *Obliczenia inteligentne, szybkie przekształcenia i klasyfikatory*, Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2004.

[138] N. Md Tahir, A. Hussain, S. Abdul Samad, H. Husain, R. A. Rahman, *Human Shape Recognition using Fourier Descriptor*, Journal of Electrical and Electronic Systems Research, wol. 2, czerwiec 2009.

[139] E. J. Tan, Z. Ignjatovic, M. F. Bocko, *A CMOS Image Sensor with Focal Plane Discrete Cosine Transform Computation*, IEEE International Symposium on Circuits and Systems, str. 2395-2398, maj 2007.

[140] G. F. Taylor, R. H. Steinvorth, J. F. McDonald, *An Architecture for a Video Rate Two-Dimensional Fast Fourier Transform Processor*, IEEE Trans. On Computers, wol. 37, nr 9, str. 1145-1148, 1988.

[141] R. Tello, *Fourier Descriptors For Computer Graphics*, IEEE Trans. on Systems. Man. and Cybernetics, wol. 25, nr 5, str. 861-865, maj 1995.

[142] Yeun-Renn Ting, Erl-Huei Lu, Chiou-Yng Lee, *A Complex-valued Cyclic Code Using Fast Fourier Transform*, IEEE, 2001.

[143] Chien-Cheng Tseng, Tsung-Ming Hwang, *Quantum Circuit Design of 8x8 Discrete Cosine Transform Using Its Fast Computation Flow Graph*, IEEE ISCAS International Symposium on Circuits and Systems, wol. 1, str. 828-831, maj 2005.

[144] Z. Wang, *Fast algorithms for the discrete W transform and for the discrete Fourier transform*, IEEE Trans. Acoust., Speech and Signal Process., wol. ASSP-32, str. 803-816, sierpień 1984.

[145] Z. Wang, *Reconsideration of a fast computational algorithm for the discrete cosine transform*, IEEE Trans. Commun., wol. COM-31, str. 121-123, styczeń 1983.

[146] P. A. Warrick, D. Precup, E. F. Hamilton, R. E. Kearney, *Fetal Heart Rate Deceleration Detection from the Discrete Cosine Transform Spectrum*, IEEE Proc. Engineering in Medicine and Biology, str. 5555-5558, wrzesień 2005.

[147] Ng. Wee, Lin Zhiping, *A New Shape-Adaptive DCT For Coding of Arbitrarily Shaped Image Segments*, IEEE ICASSP Proc. International Conference on Acoustics, Speech, and Signal Processing, wol. 6, str. 2115-2118, czerwiec 2000.

[148] W. J. Wilson, M. Winter, C.Nohr, F. Aghdas, *Signal Processing of The Auditory Brainstem, Response: Clinical Effects of Variations in Fast Fourier Transform Analysis*, IEEE, 1998.

[149] S. Winograd, *On Computing the Discrete Fourier Transform*, Math. Comput. 32, str. 175-199, 1978.

[150] J. L. Wu, S. H. Hsu, W. J. Duh, *A Novel Two-Stage Algorithm for DCT and IDCT*, IEEE Trans. on Signal Processing, wol. 39, nr 1, str. 54-61, styczeń 1992.

[151] S. Venkataraman *et al.*, *Discrete Transforms Via the Walsh-Hadamard Transform*, Proc. 26th Midwest Symp. Circuits and Systems, str. 74-78, październik 1983.

[152] M. Vetterli, H. Nussbaumer, *Simple FFT and DCT Algorithms With Reduced Number of Operations*, Signal Processing, wol. 6, str. 267-278, październik 1984.

[153] Shi Xin, Zhang Tiejun, Hou Chaohuan, *A High-Performance Power-Efficient Structure of FFT (Fast Fourier Transform) Processor*, IEEE Proc. of ICSP Conference, str. 555-558, 2004.

[154] L. P. Yaroslavsky, *Digital Picture Processing. An Introduction*, Springer-Verlag, Berlin, 1985.

[155] M. N. Yatsymirskyy, *Adaptacyjne algorytmy do obliczenia przekształcenia Fouriera*, Elektrotechnika Teoretyczna, Zeszyt. 46, str. 40-47, 1989 (język ukraiński).

[156] M. M. Yatsymirskyy, R. I. Liskevytch, *Struktury algorytmów obliczania przekształceń: Fouriera, Hartleya, kosinusowego i sinusowego*, Modelowanie i Technologie Informacyjne, nr 2, s. 173-181, Instytut Modelowania Problemów Inżynierii Energetycznej, Ukraińska Akademia Nauk, Kijów, Ukraina, 1999.

[157] P. Yip, K. R. Rao, *The Decimation-In-Frequency Algorithms For a Family of Discrete Sine And Cosine Transforms*, Circuits, Systems and Signal Processing, str. 4-19, 1988.

[158] Dengsheng Zhang, Guojun Lu, *Generic Fourier Descriptor for Shape-based Image Retrieval*, IEEE International Conference on Multimedia and Expo, wol. 1, str. 425-428, 2002.

[159] Dengsheng Zhang, Guojun Lu, *A Comparative Study of Fourier Descriptors for Shape Recognition and Retrieval*, Proc. 5th Asian Conference On Computer Vision, styczeń 2002.

[160] Z. Zhijin, *Efficient Implementation of Structures of AC-3's Filter Banks*, ICSP Proc. Fourth International Conference on Signal Processing, str. 31-34, 1998.

[161] T. P. Zieliński, *Od teorii do cyfrowego przetwarzania sygnałów*, AGH, Kraków 2002.

Spis tabel

Spis rysunków

Skorowidz

A

Przykładowe implementacje omawianych algorytmów

W niniejszym dodatku zamieszczono procedury realizujące szybkie algorytmy obliczania jedno- i dwuwymiarowych przekształceń: Fouriera oraz kosinusowych i sinusowych drugiego i czwartego rodzaju. Procedury zostały napisane w języku C z wykorzystaniem standardowej biblioteki procedur matematycznych *math.h*.

PROCEDURA A.1 *(Procedura realizująca przemieszanie bit − reverse)*
Procedura realizuje przemieszanie N-elementowej tablicy in_tab zgodnie z permutacją bit − reverse. Parametr n oznacza potęgę, do której należy podnieść liczbę dwa, ażeby uzyskać długość tablicy, tzn. $n = log_2(N)$.

```
1  void bitreverse(double* in_tab, int N, int n)
2  {
3     double a;
4     int c, t;
5     for(int i = 0;i < N;i++)
6     {
7        t = i; c = 0;
8        for(int k = 0;k < n;k++)
9        {
10          c <<= 1;
11          c += t & 1;
12          t >>= 1;
13       }
14       if (i < c)
15       {
16          a         = in_tab[i];
17          in_tab[i] = in_tab[c];
18          in_tab[c] = a;
19       }
20    }
21 }
```

PROCEDURA A.2 *(Implementacja algorytmu FFT typu radix-2)*
Jako parametry wejściowe procedura przyjmuje dwie tablice in_re oraz in_im, w których zapisane są odpowiednio części rzeczywiste i urojone elementów wejściowego ciągu danych o długości N. Parametr $n = log_2(N)$.

```
#define PI 3.141592654

void FFTradix2(double* in_re, double* in_im, int N, int n)
{
   int     l, p, p2;
   double  c, s, a, b, c1, s1;
   l  = N/2, p  = 1, p2 = 2;
   bitreverse(in_re, N, n);
   bitreverse(in_im, N, n);
   for(int i = 0;i < n;i++)
   {
      for(int j = 0;j < l;j++)
      {
         for(int k = 0;k < p;k++)
         {
            if (k != 0)
            {
               a                    = in_re[j*p2 + p + k];
               b                    = in_im[j*p2 + p + k];
               in_re[j*p2 + p + k] = a*c + b*s;
               in_im[j*p2 + p + k] = b*c - a*s;
               a                    = c*c1 - s*s1;
               s                    = s*c1 + c*s1;
               c                    = a;
            }
            else
            {
               c1                   = cos(PI/p);
               s1                   = sin(PI/p);
               c                    = c1;
               s                    = s1;
            }
            a                    = in_re[j*p2 + k];
            b                    = in_re[j*p2 + p + k];
            in_re[j*p2 + k]      = a + b;
            in_re[j*p2 + p + k]  = a - b;
            a                    = in_im[j*p2 + k];
            b                    = in_im[j*p2 + p + k];
            in_im[j*p2 + k]      = a + b;
            in_im[j*p2 + p + k]  = a - b;
         }
      }
      l  >>= 1;
      p  <<= 1;
      p2 <<= 1;
   }
}
```

PROCEDURA A.3 *(Przemieszanie dla algorytmów z przerzedzeniem w czasie) Poniższa procedura realizuje przemieszanie wejściowej N - elementowej tablicy in_tab, zgodnie z kolejnością charakterystyczną dla szybkich algorytmów z przerzedzeniem w czasie dla dyskretnego kosinusowego i sinusowego przekształcenia drugiego i czwartego rodzaju. Parametr n musi spełniać następujący warunek $2^n = N$.*

```
void pwcz(double* in_tab, int N, int n)
{
   int     m, l, k1, k2;
   double* t, *p1, *p2;
   t = new double[N];
   m = 1; l = N/2;
   for(int e = 0;e < n - 1;e++)
   {
      if (e%2 == 0)
      {
         p1 = in_tab; p2 = t;
```

```
12          }
13          else
14          {
15              p1 = t; p2 = in_tab;
16          }
17          for(int j = 0;j < m;j++)
18          {
19              for(int k = 0;k < l;k++)
20              {
21                  if (k%2 == 0)
22                  {
23                      k1 = 2*k; k2 = 2*k + 1;
24                  }
25                  else
26                  {
27                      k1 = 2*k + 1; k2 = 2*k;
28                  }
29                  p2[k + 2*l*j]     = p1[k1 + 2*l*j];
30                  p2[k + l + 2*l*j] = p1[k2 + 2*l*j];
31              }
32          }
33          m <<= 1;
34          l >>= 1;
35      }
36      if (n%2 == 0)
37      {
38          for(int k = 0;k < N;k++)
39          {
40              in_tab[k] = t[k];
41          }
42      }
43      delete[] t;
44  }
```

PROCEDURA A.4 *(Implementacja algorytmu FCT-II z przerzedzeniem w czasie) Parametry wejściowe oznaczają odpowiednio: in_tab - tablica próbek sygnału, N - długość przekształcenia oraz parametr n spełniający warunek* $2^n = N$.

```
1   #define PI2  1.570796326
2   #define SQR2 0.707106781
3
4   void FCTII(double* in_tab, int N, int n)
5   {
6       int    m, l, p;
7       double t, c, s, ca, sa, a, b;
8       pwcz(in_tab, N, n);
9       p = 2;
10      m = N/2;
11      l = 1;
12      for(int i = 0;i < n;i++)
13      {
14          for(int j = 0;j < m;j++)
15          {
16              for(int k = 0;k < l;k++)
17              {
18                  t                   = in_tab[j*p + k];
19                  in_tab[j*p + k]     = in_tab[j*p + k] + in_tab[j*p + k + l];
20                  in_tab[j*p + k + l] = t - in_tab[j*p + k + l];
21              }
22          }
23          c  = cos(PI2/p);
24          s  = sin(PI2/p);
25          ca = c; sa = s;
26          for(int k = 0;k < m;k++)
27          {
28              in_tab[k*p + l] *= SQR2;
```

```
29          }
30          for(int k = 1;k < l;k++)
31          {
32              for(int j = 0;j < m;j++)
33              {
34                  a                       = in_tab[j*p + k];
35                  b                       = in_tab[(j + 1)*p - k];
36                  in_tab[j*p + k]         = (c*a + s*b);
37                  in_tab[(j + 1)*p - k] = (c*b - s*a);
38              }
39              t = c;
40              c = (c*ca - s*sa);
41              s = (s*ca + t*sa);
42          }
43          l <<= 1;
44          m >>= 1;
45          p <<= 1;
46      }
47  }
```

PROCEDURA A.5 *(Implementacja algorytmu FCT-IV z przerzedzeniem w czasie)*
Implementacja szybkiego algorytmu z przerzedzeniem w czasie dla dyskretnego przekształcenia kosinusowego czwartego rodzaju o długości N, operującego na ciągu danych wejściowych in_tab, gdzie parametr n spełnia warunek $2^n = N$.

```
1   #define PI2  1.570796326
2   #define SQR2 0.707106781
3
4   void FCTIV(double* in_tab, int N, int n)
5   {
6       int    m, l, p;
7       double t, c, s, ca, sa, a, b;
8       pwcz(in_tab, N, n);
9       p = 2;
10      m = N/2;
11      l = 1;
12      for(int i = 0;i < N;i++)
13      {
14          in_tab[i] *= SQR2;
15      }
16      for(int i = 0;i < n;i++)
17      {
18          for(int j = 0;j < m;j++)
19          {
20              for(int k = 0;k < l;k++)
21              {
22                  t                   = in_tab[j*p + k];
23                  in_tab[j*p + k]     = in_tab[j*p + k] + in_tab[j*p + k + l];
24                  in_tab[j*p + k + l] = t - in_tab[j*p + k + l];
25              }
26          }
27          c  = cos(0.5*PI2/p);
28          s  = sin(0.5*PI2/p);
29          ca = cos(PI2/p);
30          sa = sin(PI2/p);
31          for(int k = 0;k < l;k++)
32          {
33              for(int j = 0;j < m;j++)
34              {
35                  a                         = in_tab[j*p + k];
36                  b                         = in_tab[(j + 1)*p - k - 1];
37                  in_tab[j*p + k]           = (c*a + s*b);
38                  in_tab[(j + 1)*p - k - 1] = (c*b - s*a);
39              }
```

```
40            t = c;
41            c = (c*ca - s*sa);
42            s = (s*ca + t*sa);
43         }
44         l <<= 1;
45        m >>= 1;
46         p <<= 1;
47     }
48  }
```

PROCEDURA A.6 *(Implementacja algorytmu FST-II z przerzedzeniem w czasie) Jako parametry wejściowe przyjmuje: in_tab - tablicę danych wejściowych, N - długość przekształcenia oraz parametr n spełniający następujący warunek* $2^n = N$.

```
1  #define PI2  1.570796326
2  #define SQR2 0.707106781
3
4  void FSTII(double* in_tab, int N, int n)
5  {
6     int    m, l, p;
7     double t, c, s, ca, sa, a, b;
8     pwcz(in_tab, N, n);
9     p = 2;
10    m = N/2;
11    l = 1;
12    for(int i = 0;i < n;i++)
13    {
14       for(int j = 0;j < m;j++)
15       {
16          for(int k = 0;k < l;k++)
17          {
18             t                    = in_tab[j*p + k];
19             in_tab[j*p + k]      = in_tab[j*p + k] + in_tab[j*p + k + l];
20             in_tab[j*p + k + l] = t - in_tab[j*p + k + l];
21          }
22       }
23       c  = cos(PI2/p);
24       s  = sin(PI2/p);
25       ca = c;
26       sa = s;
27       for(int k = 0;k < m;k++)
28       {
29          in_tab[k*p + l - 1] *= SQR2;
30       }
31       for(int k = 0;k < l - 1;k++)
32       {
33          for(int j = 0;j < m;j++)
34          {
35             a                          = in_tab[j*p + k];
36             b                          = in_tab[(j + 1)*p - k - 2];
37             in_tab[j*p + k]            = (c*a - s*b);
38             in_tab[(j + 1)*p - k - 2] = (s*a + c*b);
39          }
40          t = c;
41          c = (c*ca - s*sa);
42          s = (s*ca + t*sa);
43       }
44       l <<= 1;
45       m >>= 1;
46       p <<= 1;
47    }
48 }
```

PROCEDURA A.7 *(Implementacja algorytmu FST-IV z przerzedzeniem w czasie) Przyjmowane parametry wejściowe to: in_tab - tablica danych wejściowych, N - długość przekształcenia oraz parametr n, który spełnia następujący warunek* $2^n = N$.

```
#define PI2  1.570796326
#define SQR2 0.707106781

void FSTIV(double* in_tab, int N, int n)
{
   int    m, l, p;
   double t, c, s, ca, sa, a, b;
   pwcz(in_tab, N, n);
   p = 2;
   m = N/2;
   l = p >> 1;
   for(int i = 0;i < N;i++)
   {
      in_tab[i] *= SQR2;
   }
   for(int i = 0;i < n;i++)
   {
      for(int j = 0;j < m;j++)
      {
         for(int k = 0;k < l;k++)
         {
            t                   = in_tab[j*p + k];
            in_tab[j*p + k]     = in_tab[j*p + k] + in_tab[j*p + k + l];
            in_tab[j*p + k + l] = t - in_tab[j*p + k + l];
         }
      }
      c  = cos(0.5*PI2/p);
      s  = sin(0.5*PI2/p);
      ca = cos(PI2/p);
      sa = sin(PI2/p);
      for(int k = 0;k < l;k++)
      {
         for(int j = 0;j < m;j++)
         {
            a                           = in_tab[j*p + k];
            b                           = in_tab[(j + 1)*p - k - 1];
            in_tab[j*p + k]             = (c*a - s*b);
            in_tab[(j + 1)*p - k - 1] = (c*b + s*a);
         }
         t = c;
         c = (c*ca - s*sa);
         s = (s*ca + t*sa);
      }
      l <<= 1;
      m >>= 1;
      p <<= 1;
   }
}
```

PROCEDURA A.8 *(Przemieszanie bit − reverse dla dwuwymiarowej tablicy) Poniższa procedura realizuje przemieszanie dwuwymiarowej tablicy in_tab zgodnie z kolejnością bit − reverse, zastosowaną względem N wierszy i M kolumn tablicy. Parametry n i m spełniają następujące warunki* $2^n = N$ *i* $2^m = M$. *Procedura wykorzystuje procedurę sortowania* **bitreverse** *dla tablic jednowymiarowych.*

```
void bitreverse2D(double** in_tab, int N, int n, int M, int m)
{
    double* t;
    t = new double[N];
```

```
5       for(int i = 0;i < N;i++)
6       {
7          bitreverse(in_tab[i], M, m);
8       }
9       for(int j = 0;j < M;j++)
10      {
11         for(int i = 0;i < N;i++)
12         {
13            t[i] = in_tab[i][j];
14         }
15         bitreverse(t, N, n);
16         for(int i = 0;i < N;i++)
17         {
18            in_tab[i][j] = t[i];
19         }
20      }
21      delete[] t;
22   }
```

PROCEDURA A.9 *(Szybki algorytm typu radix-2 dla przekształcenia DFT2D) Poniższa procedura jest implementacją szybkiego algorytmu typu radix-2 dla dyskretnego dwuwymiarowego przekształcenia Fouriera, operującego na N na N-elementowym zbiorze danych reprezentowanym przez dwie tablice: in_re - części rzeczywiste i in_im - części urojone. Natomiast parametry n i m obliczamy odpowiednio jako* $n = log_2 N$ *i* $m = log_2 M$*. Procedura wykorzystuje dodatkowo procedurę* **FFTradix2bp***, która jest implementacją agorytmu FFT dla przekształcenia jednowymiarowego, bez przemieszania elementów wejściowej tablicy danych.*

```
1  #define PI 3.141592654
2
3  void FFT2Dradix2(double** in_re, double** in_im, int N, int n, int M, int m)
4  {
5   int     mx, ik, il, sk, sl;
6   double cok, sik, col, sil, ckl, skl;
7   double cok1, sik1, col1, sil1, ckl1, skl1;
8   double a, b, c, d, *t_re, *t_im;
9   sk = 1;    sl = 1;
10  ik = N/2;  il = M/2;
11  mx = n > m ? n : m;
12  bitreverse2D(in_re, N, M, n, m);
13  bitreverse2D(in_im, N, M, n, m);
14  if (N > M)
15  {
16   ik    = M;    il = M;
17   sk    = N/M;  sl = 1;
18   t_re = new double[sk];
19   t_im = new double[sk];
20   for(int i = 0;i < ik;i++)
21   {
22    for(int j = 0;j < il;j++)
23    {
24     for(int k = 0;k < sk;k++)
25     {
26      t_re[k] = in_re[i*sk + k][j];
27      t_im[k] = in_im[i*sk + k][j];
28     }
29     FFTradix2bp(t_re, t_im, sk, n - m);
30     for(int k = 0;k < sk;k++)
31     {
32      in_re[i*sk + k][j] = t_re[k];
33      in_im[i*sk + k][j] = t_im[k];
34     }
35    }
36   }
```

```
37     delete[] t_re;
38     delete[] t_im;
39     sk = N/M; sl = 1;
40     ik = M/2; il = M/2;
41     mx -= 1;
42    }
43    if (N < M)
44    {
45     ik   = N; il = N;
46     sk   = 1; sl = M/N;
47     t_re = new double[sl];
48     t_im = new double[sl];
49     for(int i = 0;i < ik;i++)
50     {
51      for(int j = 0;j < il;j++)
52      {
53       for(int k = 0;k < sl;k++)
54       {
55        t_re[k] = in_re[i][j*sl + k];
56        t_im[k] = in_im[i][j*sl + k];
57       }
58       FFTradix2bp(t_re, t_im, sl, m - n);
59       for(int k = 0;k < sl;k++)
60       {
61        in_re[i][j*sl + k] = t_re[k];
62        in_im[i][j*sl + k] = t_im[k];
63       }
64      }
65     }
66     delete[] t_re;
67     delete[] t_im;
68     sk = 1;    sl = M/N;
69     ik = N/2; il = N/2;
70     mx -= 1;
71    }
72    for(int e = 0;e < mx;e++)
73    {
74     for(int i = 0;i < ik;i++)
75     {
76      for(int j = 0;j < il;j++)
77      {
78       for(int k = 0;k < sk;k++)
79       {
80        for(int l = 0;l < sl;l++)
81        {
82         if (l != 0)
83         {
84          a = in_re[i*(2*sk) + k][j*(2*sl) + l + sl];
85          b = in_im[i*(2*sk) + k][j*(2*sl) + l + sl];
86          in_re[i*(2*sk) + k][j*(2*sl) + l + sl] = a*col + b*sil;
87          in_im[i*(2*sk) + k][j*(2*sl) + l + sl] = b*col - a*sil;
88         }
89         else
90         {
91          col1 = cos(PI/sl);
92          sil1 = sin(PI/sl);
93          col  = 1.0;
94          sil  = 0.0;
95         }
96         if (k != 0)
97         {
98          a = in_re[i*(2*sk) + k + sk][j*(2*sl) + l];
99          b = in_im[i*(2*sk) + k + sk][j*(2*sl) + l];
100         in_re[i*(2*sk) + k + sk][j*(2*sl) + l] = a*cok + b*sik;
101         in_im[i*(2*sk) + k + sk][j*(2*sl) + l] = b*cok - a*sik;
102        }
103        else
104        {
105         cok1 = cos(PI/sk);
106         sik1 = sin(PI/sk);
107         cok  = 1.0;
108         sik  = 0.0;
```

```
109         }
110         if (k != 0 || l != 0)
111         {
112          ckl = cok*col - sik*sil;
113          skl = sik*col + cok*sil;
114          a = in_re[i*(2*sk) + k + sk][j*(2*sl) + l + sl];
115          b = in_im[i*(2*sk) + k + sk][j*(2*sl) + l + sl];
116          in_re[i*(2*sk) + k + sk][j*(2*sl) + l + sl] = a*ckl + b*skl;
117          in_im[i*(2*sk) + k + sk][j*(2*sl) + l + sl] = b*ckl - a*skl;
118         }
119         a   = col*col1 - sil*sil1;
120         sil = sil*col1 + col*sil1;
121         col = a;
122        }
123        a   = cok*cok1 - sik*sik1;
124        sik = sik*cok1 + cok*sik1;
125        cok = a;
126       }
127      }
128     }
129     for(int i = 0;i < ik;i++)
130     {
131      for(int j = 0;j < il;j++)
132      {
133       for(int k = 0;k < sk;k++)
134       {
135        for(int l = 0;l < sl;l++)
136        {
137         a = in_re[i*(2*sk) + k][j*(2*sl) + l];
138         b = in_re[i*(2*sk) + k + sk][j*(2*sl) + l];
139         c = in_re[i*(2*sk) + k][j*(2*sl) + l + sl];
140         d = in_re[i*(2*sk) + k + sk][j*(2*sl) + l + sl];
141         in_re[i*(2*sk) + k][j*(2*sl) + l]           = (a + b + c + d);
142         in_re[i*(2*sk) + k + sk][j*(2*sl) + l]      = (a - b + c - d);
143         in_re[i*(2*sk) + k][j*(2*sl) + l + sl]      = (a + b - c - d);
144         in_re[i*(2*sk) + k + sk][j*(2*sl) + l + sl] = (a - b - c + d);
145         a = in_im[i*(2*sk) + k][j*(2*sl) + l];
146         b = in_im[i*(2*sk) + k + sk][j*(2*sl) + l];
147         c = in_im[i*(2*sk) + k][j*(2*sl) + l + sl];
148         d = in_im[i*(2*sk) + k + sk][j*(2*sl) + l + sl];
149         in_im[i*(2*sk) + k][j*(2*sl) + l]           = (a + b + c + d);
150         in_im[i*(2*sk) + k + sk][j*(2*sl) + l]      = (a - b + c - d);
151         in_im[i*(2*sk) + k][j*(2*sl) + l + sl]      = (a + b - c - d);
152         in_im[i*(2*sk) + k + sk][j*(2*sl) + l + sl] = (a - b - c + d);
153        }
154       }
155      }
156     }
157     ik = ik/2;
158     sk = sk*2;
159     il = il/2;
160     sl = sl*2;
161    }
162   }
```

PROCEDURA A.10 *(Przemieszanie z przerzedzeniem w czasie dla przypadku algorytmów dwuwymiarowych)*
Procedura realizuje przemieszanie zgodne ze schematem charakterystycznym dla dwuwymiarowych algorytmów z przerzedzeniem w czasie. Sortowane elementy zapisane są w N na M-elementowej tablicy in_tab. Parametry n i m obliczamy jako $n = \log_2 N$ i $m = \log_2 M$.

```
1  void pwcz2D(double** in_tab, int N, int n, int M, int m)
2  {
3      double* t;
```

```
 4       t = new double[N];
 5       for(int i = 0;i < N;i++)
 6       {
 7          pwcz(in_tab[i], M, m);
 8       }
 9       for(int j = 0;j < M;j++)
10       {
11          for(int i = 0;i < N;i++)
12          {
13             t[i] = in_tab[i][j];
14          }
15          pwcz(t, N, n);
16          for(int i = 0;i < N;i++)
17          {
18             in_tab[i][j] = t[i];
19          }
20       }
21       delete[] t;
22    }
```

PROCEDURA A.11 *(Szybki algorytm z przerzedzeniem w czasie dla DCT2D-II) Procedura jest implemenacją szybkiego algorytmu z przerzedzeniem w czasie dla przekształcenia DCT2D-II. Na wejściu wymagana jest dwuwymiarowa tablica próbek sygnału in_tab, liczba próbek N na M oraz parametry $n = log_2 N$ i $m = log_2 M$. Procedura korzysta z szybkiego algorytmu (procedura* **FCTIIbp***) dla jednowymiarowego przekształcenia DCT-II, bez przemieszania elementów tablicy wejściowej.*

```
 1  #define PI 3.141592654
 2  #define SQR2 0.707106781
 3
 4  void FCT2DII(double** in_tab, int N, int n, int M, int m)
 5  {
 6   int    mx, ik, il, sk, sl;
 7   double c1k, cok, s1k, sik;
 8   double c1l, col, s1l, sil;
 9   double a, b, c, d, *t;
10   sk = 1;    sl = 1;
11   ik = N/2; il = M/2;
12   mx = n > m ? n : m;
13   pwcz2D(in_tab, N, n, M, m);
14   if (N > M)
15   {
16    ik = M;    il = M;
17    sk = N/M; sl = 1;
18    t  = new double[sk];
19    for(int i = 0;i < ik;i++)
20    {
21     for(int j = 0;j < il;j++)
22     {
23      for(int k = 0;k < sk;k++)
24      {
25       t[k] = in_tab[i*sk + k][j];
26      }
27      FCTIIbp(t, sk, n - m);
28      for(int k = 0;k < sk;k++)
29      {
30       in_tab[i*sk + k][j] = t[k];
31      }
32     }
33    }
34    delete[] t;
35    sk = N/M; sl = 1;
36    ik = M/2; il = M/2;
37    mx -= 1;
38   }
```

```
39   if (N < M)
40   {
41    ik = N; il = N;
42    sk = 1; sl = M/N;
43    t  = new double[sl];
44    for(int i = 0;i < ik;i++)
45    {
46     for(int j = 0;j < il;j++)
47     {
48      for(int k = 0;k < sl;k++)
49      {
50       t[k] = in_tab[i][j*sl + k];
51      }
52      FCTIIbp(t, sl, m - n);
53      for(int k = 0;k < sl;k++)
54      {
55       in_tab[i][j*sl + k] = t[k];
56      }
57     }
58    }
59    delete[] t;
60    sk = 1;    sl = M/N;
61    ik = N/2; il = N/2;
62    mx -= 1;
63   }
64   for(int e = 0;e < mx;e++)
65   {
66    for(int i = 0;i < ik;i++)
67    {
68     for(int j = 0;j < il;j++)
69     {
70      for(int k = 0;k < sk;k++)
71      {
72       for(int l = 0;l < sl;l++)
73       {
74        a = in_tab[i*(2*sk) + k][j*(2*sl) + l];
75        b = in_tab[i*(2*sk) + k + sk][j*(2*sl) + l];
76        c = in_tab[i*(2*sk) + k][j*(2*sl) + l + sl];
77        d = in_tab[i*(2*sk) + k + sk][j*(2*sl) + l + sl];
78        in_tab[i*(2*sk) + k][j*(2*sl) + l]           = a + b + c + d;
79        in_tab[i*(2*sk) + k + sk][j*(2*sl) + l]      = a - b + c - d;
80        in_tab[i*(2*sk) + k][j*(2*sl) + l + sl]      = a + b - c - d;
81        in_tab[i*(2*sk) + k + sk][j*(2*sl) + l + sl] = a - b - c + d;
82       }
83      }
84     }
85    }
86    c1k = cos(PI/(4.0*sk));
87    s1k = sin(PI/(4.0*sk));
88    cok = c1k; sik = s1k;
89    for(int k = 1;k < sk;k++)
90    {
91     c1l = cos(PI/(4.0*sl));
92     s1l = sin(PI/(4.0*sl));
93     col = c1l; sil = s1l;
94     for(int l = 1;l < sl;l++)
95     {
96      for(int i = 0;i < ik;i++)
97      {
98       for(int j = 0;j < il;j++)
99       {
100       a   = in_tab[i*(2*sk) + k][j*(2*sl) + l];
101       b   = in_tab[(i+1)*(2*sk) - k][j*(2*sl) + l];
102       c   = in_tab[i*(2*sk) + k][(j+1)*(2*sl) - l];
103       d   = in_tab[(i+1)*(2*sk) - k][(j+1)*(2*sl) - l];
104       in_tab[i*(2*sk) + k][j*(2*sl) + l]         =  a*cok*col + b*sik*col
105                                                  + c*cok*sil + d*sik*sil;
106       in_tab[(i+1)*(2*sk) - k][j*(2*sl) + l]     = -a*sik*col + b*cok*col
107                                                  - c*sik*sil + d*cok*sil;
108       in_tab[i*(2*sk) + k][(j+1)*(2*sl) - l]     = -a*cok*sil - b*sik*sil
109                                                  + c*cok*col + d*sik*col;
110       in_tab[(i+1)*(2*sk) - k][(j+1)*(2*sl) - l] =  a*sik*sil - b*cok*sil
```

```
111                                                    - c*sik*col + d*cok*col;
112       }
113      }
114      a   = c1l*col - s1l*sil;
115      sil = s1l*col + c1l*sil;
116      col = a;
117     }
118     a   = c1k*cok - s1k*sik;
119     sik = s1k*cok + c1k*sik;
120     cok = a;
121    }
122    for(int i = 0;i < ik;i++)
123    {
124     for(int j = 0;j < il;j++)
125     {
126      in_tab[i*(2*sk) + sk][j*(2*sl)]       *= SQR2;
127      in_tab[i*(2*sk)][j*(2*sl) + sl]       *= SQR2;
128      in_tab[i*(2*sk) + sk][j*(2*sl) + sl] *= 0.5;
129     }
130    }
131    c1k = cos(PI/(4.0*sk));
132    s1k = sin(PI/(4.0*sk));
133    cok = c1k; sik = s1k;
134    for(int k = 1;k < sk;k++)
135    {
136      for(int i = 0;i < ik;i++)
137      {
138       for(int j = 0;j < il;j++)
139       {
140        a   = in_tab[i*(2*sk) + k][j*(2*sl)];
141        b   = in_tab[(i+1)*(2*sk) - k][j*(2*sl)];
142        in_tab[i*(2*sk) + k][j*(2*sl)]          =  a*cok + b*sik;
143        in_tab[(i+1)*(2*sk) - k][j*(2*sl)]      = -a*sik + b*cok;
144        a   = in_tab[i*(2*sk) + k][j*(2*sl) + sl];
145        b   = in_tab[(i+1)*(2*sk) - k][j*(2*sl) + sl];
146        in_tab[i*(2*sk) + k][j*(2*sl) + sl]     = SQR2*(a*cok + b*sik);
147        in_tab[(i+1)*(2*sk) - k][j*(2*sl) + sl] = SQR2*(-a*sik + b*cok);
148       }
149      }
150      a   = c1k*cok - s1k*sik;
151      sik = s1k*cok + c1k*sik;
152      cok = a;
153     }
154     c1l = cos(PI/(4.0*sl));
155     s1l = sin(PI/(4.0*sl));
156     col = c1l; sil = s1l;
157     for(int l = 1;l < sl;l++)
158     {
159       for(int i = 0;i < ik;i++)
160       {
161        for(int j = 0;j < il;j++)
162        {
163         a   = in_tab[i*(2*sk)][j*(2*sl) + l];
164         b   = in_tab[i*(2*sk)][(j+1)*(2*sl) - l];
165         in_tab[i*(2*sk)][j*(2*sl) + l]          =  a*col + b*sil;
166         in_tab[i*(2*sk)][(j+1)*(2*sl) - l]      = -a*sil + b*col;
167         a   = in_tab[i*(2*sk) + sk][j*(2*sl) + l];
168         b   = in_tab[i*(2*sk) + sk][(j+1)*(2*sl) - l];
169         in_tab[i*(2*sk) + sk][j*(2*sl) + l]     = SQR2*(a*col + b*sil);
170         in_tab[i*(2*sk) + sk][(j+1)*(2*sl) - l] = SQR2*(-a*sil + b*col);
171        }
172       }
173       a   = c1l*col - s1l*sil;
174       sil = s1l*col + c1l*sil;
175       col = a;
176     }
177     ik >>= 1; il >>= 1;
178     sk <<= 1; sl <<= 1;
179    }
180   }
```

PROCEDURA A.12 *(Szybki algorytm z przerzedzeniem w czasie dla DCT2D-IV) Procedura implementująca szybki algorytm z przerzedzeniem w czasie dla przekształcenia DCT2D-IV. Procedura przyjmuje następujące argumenty: dwuwymiarową tablicę próbek sygnału in_tab, ilość N na M próbek oraz parametry spełniające warunki* $n = log_2 N$ *i* $m = log_2 M$*. Procedura korzysta z algorytmu FCT-IV (***FCTIVbp***) bez wstępnego przemieszania elementów wejściowej tablicy danych.*

```
#define PI 3.141592654
#define SQR2 0.707106781

void FCT2DIV(double** in_tab, int N, int n, int M, int m)
{
 int    mx, ik, il, sk, sl;
 double c1k, cok, s1k, sik;
 double c1l, col, s1l, sil;
 double a, b, c, d, *t;
 sk = 1;   sl = 1;
 ik = N/2; il = M/2;
 mx = n > m ? n : m;
 pwcz2D(in_tab, N, n, M, m);
 if (N > M)
 {
  ik = M;    il = M;
  sk = N/M; sl = 1;
  t  = new double[sk];
  for(int i = 0;i < ik;i++)
  {
   for(int j = 0;j < il;j++)
   {
    for(int k = 0;k < sk;k++)
    {
     t[k] = in_tab[i*sk + k][j];
    }
    FCTIVbp(t, sk, n - m);
    for(int k = 0;k < sk;k++)
    {
     in_tab[i*sk + k][j] = SQR2*t[k];
    }
   }
  }
  delete[] t;
  sk = N/M; sl = 1;
  ik = M/2; il = M/2;
  mx -= 1;
 }
 if (N < M)
 {
  ik = N; il = N;
  sk = 1; sl = M/N;
  t  = new double[sl];
  for(int i = 0;i < ik;i++)
  {
   for(int j = 0;j < il;j++)
   {
    for(int k = 0;k < sl;k++)
    {
     t[k] = in_tab[i][j*sl + k];
    }
    FCTIVbp(t, sl, m - n);
    for(int k = 0;k < sl;k++)
    {
     in_tab[i][j*sl + k] = SQR2*t[k];
    }
   }
  }
  delete[] t;
  sk = 1;   sl = M/N;
  ik = N/2; il = N/2;
```

```
62     mx -= 1;
63    }
64    if (N == M)
65    {
66     for(int i = 0;i < N;i++)
67     {
68      for(int j = 0;j < M;j++)
69      {
70       in_tab[i][j] *= 0.5;
71      }
72     }
73    }
74    for(int e = 0;e < mx;e++)
75    {
76     for(int i = 0;i < ik;i++)
77     {
78      for(int j = 0;j < il;j++)
79      {
80       for(int k = 0;k < sk;k++)
81       {
82        for(int l = 0;l < sl;l++)
83        {
84         a = in_tab[i*(2*sk) + k][j*(2*sl) + l];
85         b = in_tab[i*(2*sk) + k + sk][j*(2*sl) + l];
86         c = in_tab[i*(2*sk) + k][j*(2*sl) + l + sl];
87         d = in_tab[i*(2*sk) + k + sk][j*(2*sl) + l + sl];
88         in_tab[i*(2*sk) + k][j*(2*sl) + l]           = a + b + c + d;
89         in_tab[i*(2*sk) + k + sk][j*(2*sl) + l]      = a - b + c - d;
90         in_tab[i*(2*sk) + k][j*(2*sl) + l + sl]      = a + b - c - d;
91         in_tab[i*(2*sk) + k + sk][j*(2*sl) + l + sl] = a - b - c + d;
92        }
93       }
94      }
95     }
96     c1k = cos(PI/(4.0*sk));
97     s1k = sin(PI/(4.0*sk));
98     cok = cos(PI/(8.0*sk));
99     sik = sin(PI/(8.0*sk));
100    for(int k = 0;k < sk;k++)
101    {
102     c1l = cos(PI/(4.0*sl));
103     s1l = sin(PI/(4.0*sl));
104     col = cos(PI/(8.0*sl));
105     sil = sin(PI/(8.0*sl));
106     for(int l = 0;l < sl;l++)
107     {
108      for(int i = 0;i < ik;i++)
109      {
110       for(int j = 0;j < il;j++)
111       {
112        a = in_tab[i*(2*sk) + k][j*(2*sl) + l];
113        b = in_tab[(i+1)*(2*sk) - k - 1][j*(2*sl) + l];
114        c = in_tab[i*(2*sk) + k][(j+1)*(2*sl) - l - 1];
115        d = in_tab[(i+1)*(2*sk) - k - 1][(j+1)*(2*sl) - l - 1];
116        in_tab[i*(2*sk) + k][j*(2*sl)+l]           =  a*cok*col + b*sik*col
117                                                   + c*cok*sil + d*sik*sil;
118        in_tab[(i+1)*(2*sk)-k-1][j*(2*sl)+l]       = -a*sik*col + b*cok*col
119                                                   - c*sik*sil + d*cok*sil;
120        in_tab[i*(2*sk)+k][(j+1)*(2*sl)-l-1]       = -a*cok*sil - b*sik*sil
121                                                   + c*cok*col + d*sik*col;
122        in_tab[(i+1)*(2*sk)-k-1][(j+1)*(2*sl)-l-1] = a*sik*sil - b*cok*sil
123                                                   - c*sik*col + d*cok*col;
124       }
125      }
126      a   = c1l*col - s1l*sil;
127      sil = s1l*col + c1l*sil;
128      col = a;
129     }
130     a   = c1k*cok - s1k*sik;
131     sik = s1k*cok + c1k*sik;
132     cok = a;
133    }
```

```
134     ik >>= 1;
135     sk <<= 1;
136     il >>= 1;
137     sl <<= 1;
138    }
139   }
```

PROCEDURA A.13 *(Szybki algorytm z przerzedzeniem w czasie dla DST2D-II) Procedura zawiera implementację szybkiego algorytmu z przerzedzeniem w czasie dla dwuwymiarowego dyskretnego przekształcenia sinusowego drugiego rodzaju. Procedura wymaga podania na wejściu dwuwymiarowej tablicy próbek sygnału in_tab, liczby N na M próbek oraz parametrów $n = log_2 N$ i $m = log_2 M$. Procedura korzysta z algorytmu FST-II bez przemieszania elementów (poprzez wywołanie procedury* **FSTIIbp***).*

```
 1  #define PI 3.141592654
 2  #define SQR2 0.707106781
 3
 4  void FST2DII(double** in_tab, int N, int n, int M, int m)
 5  {
 6   int     mx, ik, il, sk, sl;
 7   double c1k, cok, s1k, sik;
 8   double c1l, col, s1l, sil;
 9   double a, b, c, d, *t;
10   sk = 1;    sl = 1;
11   ik = N/2; il = M/2;
12   mx = n > m ? n : m;
13   pwcz2D(in_tab, N, n, M, m);
14   if (N > M)
15   {
16    ik = M;    il = M;
17    sk = N/M; sl = 1;
18    t  = new double[sk];
19    for(int i = 0;i < ik;i++)
20    {
21     for(int j = 0;j < il;j++)
22     {
23      for(int k = 0;k < sk;k++)
24      {
25       t[k] = in_tab[i*sk + k][j];
26      }
27      FSTIIbp(t, sk, n - m);
28      for(int k = 0;k < sk;k++)
29      {
30       in_tab[i*sk + k][j] = t[k];
31      }
32     }
33    }
34    delete[] t;
35    sk = N/M; sl = 1;
36    ik = M/2; il = M/2;
37    mx -= 1;
38   }
39   if (N < M)
40   {
41    ik = N; il = N;
42    sk = 1; sl = M/N;
43    t  = new double[sl];
44    for(int i = 0;i < ik;i++)
45    {
46     for(int j = 0;j < il;j++)
47     {
48      for(int k = 0;k < sl;k++)
49      {
50       t[k] = in_tab[i][j*sl + k];
51      }
```

```
52        FSTIIbp(t, sl, m - n);
53        for(int k = 0;k < sl;k++)
54        {
55         in_tab[i][j*sl + k] = t[k];
56        }
57       }
58      }
59      delete[] t;
60      sk = 1;    sl = M/N;
61      ik = N/2; il = N/2;
62      mx -= 1;
63     }
64     for(int e = 0;e < mx;e++)
65     {
66      for(int i = 0;i < ik;i++)
67      {
68       for(int j = 0;j < il;j++)
69       {
70        for(int k = 0;k < sk;k++)
71        {
72         for(int l = 0;l < sl;l++)
73         {
74          a = in_tab[i*(2*sk) + k][j*(2*sl) + l];
75          b = in_tab[i*(2*sk) + k + sk][j*(2*sl) + l];
76          c = in_tab[i*(2*sk) + k][j*(2*sl) + l + sl];
77          d = in_tab[i*(2*sk) + k + sk][j*(2*sl) + l + sl];
78          in_tab[i*(2*sk) + k][j*(2*sl) + l]           = a + b + c + d;
79          in_tab[i*(2*sk) + k + sk][j*(2*sl) + l]      = a - b + c - d;
80          in_tab[i*(2*sk) + k][j*(2*sl) + l + sl]      = a + b - c - d;
81          in_tab[i*(2*sk) + k + sk][j*(2*sl) + l + sl] = a - b - c + d;
82         }
83        }
84       }
85      }
86      c1k = cos(PI/(4.0*sk));
87      s1k = sin(PI/(4.0*sk));
88      cok = c1k; sik = s1k;
89      for(int k = 0;k < sk - 1;k++)
90      {
91       c1l = cos(PI/(4.0*sl));
92       s1l = sin(PI/(4.0*sl));
93       col = c1l; sil = s1l;
94       for(int l = 0;l < sl - 1;l++)
95       {
96        for(int i = 0;i < ik;i++)
97        {
98         for(int j = 0;j < il;j++)
99         {
100         a   = in_tab[i*(2*sk) + k][j*(2*sl) + l];
101         b   = in_tab[(i+1)*(2*sk) - k - 2][j*(2*sl) + l];
102         c   = in_tab[i*(2*sk) + k][(j+1)*(2*sl) - l - 2];
103         d   = in_tab[(i+1)*(2*sk) - k - 2][(j+1)*(2*sl) - l - 2];
104         in_tab[i*(2*sk) + k][j*(2*sl) + l]          =  a*cok*col - b*sik*col
105                                                     - c*cok*sil + d*sik*sil;
106         in_tab[(i+1)*(2*sk)-k-2][j*(2*sl)+l]        =  a*sik*col + b*cok*col
107                                                     - c*sik*sil - d*cok*sil;
108         in_tab[i*(2*sk) + k][(j+1)*(2*sl)-l-2]      =  a*cok*sil - b*sik*sil
109                                                     + c*cok*col - d*sik*col;
110         in_tab[(i+1)*(2*sk)-k-2][(j+1)*(2*sl)-l-2] =  a*sik*sil + b*cok*sil
111                                                     + c*sik*col + d*cok*col;
112        }
113       }
114       a   = c1l*col - s1l*sil;
115       sil = s1l*col + c1l*sil;
116       col = a;
117      }
118      a   = c1k*cok - s1k*sik;
119      sik = s1k*cok + c1k*sik;
120      cok = a;
121     }
122     for(int i = 0;i < ik;i++)
123     {
```

```
124      for(int j = 0;j < il;j++)
125      {
126       in_tab[(i + 1)*(2*sk) - 1][j*(2*sl) + sl - 1] *= SQR2;
127       in_tab[i*(2*sk) + sk - 1][(j + 1)*(2*sl) - 1] *= SQR2;
128       in_tab[i*(2*sk) + sk - 1][j*(2*sl) + sl - 1]  *= 0.5;
129      }
130     }
131     c1k = cos(PI/(4.0*sk));
132     s1k = sin(PI/(4.0*sk));
133     cok = c1k; sik = s1k;
134     for(int k = 0;k < sk - 1;k++)
135     {
136       for(int i = 0;i < ik;i++)
137       {
138        for(int j = 0;j < il;j++)
139        {
140         a   = in_tab[i*(2*sk)+k][j*(2*sl)+sl-1];
141         b   = in_tab[(i+1)*(2*sk)-k-2][j*(2*sl)+sl-1];
142         in_tab[i*(2*sk)+k][j*(2*sl)+sl-1]         = SQR2*(a*cok - b*sik);
143         in_tab[(i+1)*(2*sk)-k-2][j*(2*sl)+sl-1]   = SQR2*(a*sik + b*cok);
144         a   = in_tab[i*(2*sk)+k][(j+1)*(2*sl)-1];
145         b   = in_tab[(i+1)*(2*sk)-k-2][(j+1)*(2*sl)-1];
146         in_tab[i*(2*sk)+k][(j+1)*(2*sl)-1]        = a*cok - b*sik;
147         in_tab[(i+1)*(2*sk)-k-2][(j+1)*(2*sl)-1] = a*sik + b*cok;
148        }
149       }
150       a   = c1k*cok - s1k*sik;
151       sik = s1k*cok + c1k*sik;
152       cok = a;
153     }
154     c1l = cos(PI/(4.0*sl));
155     s1l = sin(PI/(4.0*sl));
156     col = c1l; sil = s1l;
157     for(int l = 0;l < sl - 1;l++)
158     {
159       for(int i = 0;i < ik;i++)
160       {
161        for(int j = 0;j < il;j++)
162        {
163         a   = in_tab[i*(2*sk)+ sk-1][j*(2*sl)+1];
164         b   = in_tab[i*(2*sk)+ sk-1][(j+1)*(2*sl)-l-2];
165         in_tab[i*(2*sk)+sk-1][j*(2*sl)+1]         = SQR2*(a*col - b*sil);
166         in_tab[i*(2*sk)+sk-1][(j+1)*(2*sl)-l-2]   = SQR2*(a*sil + b*col);
167         a   = in_tab[(i+1)*(2*sk)-1][j*(2*sl)+1];
168         b   = in_tab[(i+1)*(2*sk)-1][(j+1)*(2*sl)-l-2];
169         in_tab[(i+1)*(2*sk)-1][j*(2*sl)+1]        = a*col - b*sil;
170         in_tab[(i+1)*(2*sk)-1][(j+1)*(2*sl)-l-2] = a*sil + b*col;
171        }
172       }
173       a   = c1l*col - s1l*sil;
174       sil = s1l*col + c1l*sil;
175       col = a;
176     }
177     ik >>= 1;
178     sk <<= 1;
179     il >>= 1;
180     sl <<= 1;
181    }
182   }
```

PROCEDURA A.14 *(Szybki algorytm z przerzedzeniem w czasie dla DST2D-IV) Implementacja szybkiego algorytmu z przerzedzeniem w czasie dla przypadku przekształcenia DST2D-IV. Jako argumenty wejściowe procedura przyjmuje dwuwymiarową tablicę próbek in_tab, liczbę N na M próbek oraz parametry* $n = log_2 N$ *i* $m = log_2 M$*. Procedura korzysta z algorytmu FST-IV bez przemieszania elementów wejściowej tablicy danych (procedura* **FSTIVbp***).*

```
1  #define PI 3.141592654
2  #define SQR2 0.707106781
3
4  void FST2DIV(double** in_tab, int N, int n, int M, int m)
5  {
6   int     mx, ik, il, sk, sl;
7   double c1k, cok, s1k, sik;
8   double c1l, col, s1l, sil;
9   double a, b, c, d, *t;
10  sk = 1;    sl = 1;
11  ik = N/2; il = M/2;
12  mx = n > m ? n : m;
13  pwcz2D(in_tab, N, n, M, m);
14  if (N > M)
15  {
16   ik = M;    il = M;
17   sk = N/M; sl = 1;
18   t  = new double[sk];
19   for(int i = 0;i < ik;i++)
20   {
21    for(int j = 0;j < il;j++)
22    {
23     for(int k = 0;k < sk;k++)
24     {
25      t[k] = in_tab[i*sk + k][j];
26     }
27     FSTIVbp(t, sk, n - m);
28     for(int k = 0;k < sk;k++)
29     {
30      in_tab[i*sk + k][j] = SQR2*t[k];
31     }
32    }
33   }
34   delete[] t;
35   sk = N/M; sl = 1;
36   ik = M/2; il = M/2;
37   mx -= 1;
38  }
39  if (N < M)
40  {
41   ik = N; il = N;
42   sk = 1; sl = M/N;
43   t  = new double[sl];
44   for(int i = 0;i < ik;i++)
45   {
46    for(int j = 0;j < il;j++)
47    {
48     for(int k = 0;k < sl;k++)
49     {
50      t[k] = in_tab[i][j*sl + k];
51     }
52     FSTIVbp(t, sl, m - n);
53     for(int k = 0;k < sl;k++)
54     {
55      in_tab[i][j*sl + k] = SQR2*t[k];
56     }
57    }
58   }
59   delete[] t;
60   sk = 1;    sl = M/N;
61   ik = N/2; il = N/2;
62   mx -= 1;
63  }
64  if (N == M)
65  {
66   for(int i = 0;i < N;i++)
67   {
68    for(int j = 0;j < M;j++)
69    {
70     in_tab[i][j] *= 0.5;
71    }
```

```
72    }
73   }
74   for(int e = 0;e < mx;e++)
75   {
76    for(int i = 0;i < ik;i++)
77    {
78     for(int j = 0;j < il;j++)
79     {
80      for(int k = 0;k < sk;k++)
81      {
82       for(int l = 0;l < sl;l++)
83       {
84        a = in_tab[i*(2*sk) + k][j*(2*sl) + l];
85        b = in_tab[i*(2*sk) + k + sk][j*(2*sl) + l];
86        c = in_tab[i*(2*sk) + k][j*(2*sl) + l + sl];
87        d = in_tab[i*(2*sk) + k + sk][j*(2*sl) + l + sl];
88        in_tab[i*(2*sk) + k][j*(2*sl) + l]           = a + b + c + d;
89        in_tab[i*(2*sk) + k + sk][j*(2*sl) + l]      = a - b + c - d;
90        in_tab[i*(2*sk) + k][j*(2*sl) + l + sl]      = a + b - c - d;
91        in_tab[i*(2*sk) + k + sk][j*(2*sl) + l + sl] = a - b - c + d;
92       }
93      }
94     }
95    }
96    c1k = cos(PI/(4.0*sk));
97    s1k = sin(PI/(4.0*sk));
98    cok = cos(PI/(8.0*sk));
99    sik = sin(PI/(8.0*sk));
100   for(int k = 0;k < sk;k++)
101   {
102    c1l = cos(PI/(4.0*sl));
103    s1l = sin(PI/(4.0*sl));
104    col = cos(PI/(8.0*sl));
105    sil = sin(PI/(8.0*sl));
106    for(int l = 0;l < sl;l++)
107    {
108     for(int i = 0;i < ik;i++)
109     {
110      for(int j = 0;j < il;j++)
111      {
112       a = in_tab[i*(2*sk) + k][j*(2*sl) + l];
113       b = in_tab[(i+1)*(2*sk) - k - 1][j*(2*sl) + l];
114       c = in_tab[i*(2*sk) + k][(j+1)*(2*sl) - l - 1];
115       d = in_tab[(i+1)*(2*sk) - k - 1][(j+1)*(2*sl) - l - 1];
116       in_tab[i*(2*sk)+k][j*(2*sl)+l]             = a*cok*col - b*sik*col
117                                                  - c*cok*sil + d*sik*sil;
118       in_tab[(i+1)*(2*sk)-k-1][j*(2*sl)+l]       = a*sik*col + b*cok*col
119                                                  - c*sik*sil - d*cok*sil;
120       in_tab[i*(2*sk)+k][(j+1)*(2*sl)-l-1]       = a*cok*sil - b*sik*sil
121                                                  + c*cok*col - d*sik*col;
122       in_tab[(i+1)*(2*sk)-k-1][(j+1)*(2*sl)-l-1] = a*sik*sil + b*cok*sil
123                                                  + c*sik*col + d*cok*col;
124      }
125     }
126     a   = c1l*col - s1l*sil;
127     sil = s1l*col + c1l*sil;
128     col = a;
129    }
130    a   = c1k*cok - s1k*sik;
131    sik = s1k*cok + c1k*sik;
132    cok = a;
133   }
134   ik >>= 1;
135   sk <<= 1;
136   il >>= 1;
137   sl <<= 1;
138  }
139 }
```

PROCEDURA A.15 *(Szybki algorytm adaptacyjny dla przekształcenia Fouriera) Implementacja szybkiego algorytmu adaptacyjnego dla przekształcenia Fouriera. Na wejściu procedura wymaga podania zbioru próbek analizowanego sygnału w postaci tablic in_re i in_im, liczby próbek N oraz liczby M współczynników spektralnych branych pod uwagę w procesie adaptacji. Procedura ta odwołuje się do dwóch procedur pomocniczych* **FFTblok** *i* **FFTkrok**, *które stanowią implementacje bloków FFT oraz FD dla szybkiego algorytmu z przerzedzeniem w czasie, tutaj typu radix-2.*

```
#define PI 3.1415926

void FFTblok(double* in_re, double* in_im, int N)
{
     int     l, n, b;
     double  c, s, aa, bb;
     n = N/2;
     b = 1;
     l = (int)ceil(log(N)/log(2.0));
     for(int i = 0;i < l;i++)
     {
          for(int j = 0;j < n;j++)
          {
               for(int k = 0;k < b;k++)
               {
                    c                       = cos(PI*k/(double)b);
                    s                       = sin(PI*k/(double)b);
                    aa                      = in_re[2*j*b + b + k];
                    bb                      = in_im[2*j*b + b + k];
                    in_re[2*j*b + b + k]    = aa*c + bb*s;
                    in_im[2*j*b + b + k]    = bb*c - aa*s;
                    aa                      = in_re[2*j*b + k];
                    bb                      = in_re[2*j*b + b + k];
                    in_re[2*j*b + k]        = aa + bb;
                    in_re[2*j*b + b + k]    = aa - bb;
                    aa                      = in_im[2*j*b + k];
                    bb                      = in_im[2*j*b + b + k];
                    in_im[2*j*b + k]        = aa + bb;
                    in_im[2*j*b + b + k]    = aa - bb;
               }
          }
          n >>= 1;
          b <<= 1;
     }
}

void FFTkrok(double* in_re, double* in_im, int N)
{
     double  c, s, aa, bb;
     int b = N/2;
     for(int k = 0;k < b;k++)
     {
          c              = cos(PI*k/(double)b);
          s              = sin(PI*k/(double)b);
          aa             = in_re[b + k];
          bb             = in_im[b + k];
          in_re[b + k]   = aa*c + bb*s;
          in_im[b + k]   = bb*c - aa*s;
          aa             = in_re[k];
          bb             = in_re[b + k];
          in_re[k]       = aa + bb;
          in_re[b + k]   = aa - bb;
          aa             = in_im[k];
          bb             = in_im[b + k];
          in_im[k]       = aa + bb;
          in_im[b + k]   = aa - bb;
     }
}

void AFFT(double* in_re, double* in_im, int N, int M)
{
```

```
62      int     N1, l;
63      double* tmp_re,* tmp_im, epsmd, epso;
64      l       = (int)ceil(log(N)/log(2.0));
65      tmp_re  = new double[N];
66      tmp_im  = new double[N];
67      bitreverse(in_re, N, l);
68      bitreverse(in_im, N, l);
69      N1      = M;
70      FFTblok(N1, in_re, in_im);
71      FFTblok(N1, in_re + N1, in_im + N1);
72      N1     *= 2;
73      do
74      {
75          for(int i = 0;i < M;i++)
76          {
77              tmp_re[i] = in_re[i];
78              tmp_im[i] = in_im[i];
79          }
80          FFTkrok(in_re, in_im, N1);
81          // Oszacowanie błędu w metryce Czebyszewa
82          epsmd = 0.0;
83          for(int i = 0;i < S/2;i++)
84          {
85              epso = sqrt(pow(in_re[i] - 2.0*tab_re[i], 2.0)/(4.0*N1*N1)
86                  + pow(in_im[i] - 2.0*tab_im[i], 2.0)/(4.0*N1*N1))/3.0;
87              if (epso > epsmd)
88              {
89                  epsmd = epso;
90              }
91          }
92          if (epsmd < epsilon)
93          {
94              break;
95          }
96          if (N1 == N)
97          {
98              break;
99          }
100         FFTblok(N1, in_re + N1, in_im + N1);
101         N1 *= 2;
102     }
103     while(1);
104     // Wypisz odpowiednie informacje wg. punktu 8
105     // szybkiego algorytmu adaptacyjnego
106     delete[] tmp_re;
107     delete[] tmp_im;
108 }
```

PROCEDURA A.16 *(Szybki algorytm adaptacyjny dla przekształcenia DCT-II)* *Procedura implementująca szybki algorytm adaptacyjny dla przekształcenia DCT-II. Jako argumenty wejściowe przyjmuje tablicę próbek in_tab sygnału, liczbę N próbek, liczbę M współczynników spektralnych branych pod uwagę w procesie adaptacji, oraz dopuszczalną wartość błędu ϵ. Procedura korzysta z dwóch procedur* **FCTIIblok()** *oraz* **FCTIIkrok()**, *które stanowią odpowiednio implementacje bloków FCT i FD szybkiego algorytmu z przerzedzeniem w czasie.*

```
1  #define SQR2 0.707106781
2  #define PI2  1.570796326
3
4  void FCTIIblok(int N, double* d)
5  {
6      double t, c, s, ca, sa, a, b;
7      int    m, n, p, l;
8      l      = (int)ceil(log(N)/log(2.0));
```

```
 9       p       = 2;
10       m       = N/2;
11       n       = 1;
12       for(int i = 0;i < l;i++)
13       {
14            for(int j = 0;j < m;j++)
15            {
16                 for(int k = 0;k < n;k++)
17                 {
18                      t               = d[j*p + k];
19                      d[j*p + k]      = d[j*p + k] + d[j*p + k + n];
20                      d[j*p + k + n] = t - d[j*p + k + n];
21                 }
22            }
23            c  = cos(PI2/(double)p);
24            s  = sin(PI2/(double)p);
25            ca = c;
26            sa = s;
27            for(int k = 0;k < m;k++)
28            {
29                 d[k*p + n] *= SQR2;
30            }
31            for(int k = 1;k < n;k++)
32            {
33                 for(int j = 0;j < m;j++)
34                 {
35                      a                 = d[j*p + k];
36                      b                 = d[(j + 1)*p - k];
37                      d[j*p + k]        = (c*a + s*b);
38                      d[(j + 1)*p - k] = (c*b - s*a);
39                 }
40                 t = c;
41                 c = (c*ca - s*sa);
42                 s = (s*ca + t*sa);
43            }
44            n <<= 1;
45            m >>= 1;
46            p <<= 1;
47       }
48  }
49
50  void FCTIIkrok(int N, double* d)
51  {
52       double t, c, s, ca, sa, a, b;
53       int    n;
54       n      = N/2;
55       for(int k = 0;k < n;k++)
56       {
57            t         = d[k];
58            d[k]      = d[k] + d[k + n];
59            d[k + n] = t - d[k + n];
60       }
61       c      = cos(PI2/(double)N);
62       s      = sin(PI2/(double)N);
63       ca     = c;
64       sa     = s;
65       d[n]  *= SQR2;
66       for(int k = 1;k < n;k++)
67       {
68            a         = d[k];
69            b         = d[N - k];
70            d[k]      = (c*a + s*b);
71            d[N - k] = (c*b - s*a);
72            t         = c;
73            c         = (c*ca - s*sa);
74            s         = (s*ca + t*sa);
75       }
76  }
77
78  void AFCTII(double* in_tab, int M, int N, double epsilon)
79  {
80       int     N1, l;
```

```
81        double* tmp, epsmd, epso;
82        l       = (int)ceil(log(N)/log(2.0));
83        tmp     = new double[M];
84        pwcz(in_tab, N, l);
85        N1      = M;
86        FCTIIblok(N1, in_tab);
87        FCTIIblok(N1, in_tab + N1);
88        N1      *= 2;
89        do
90        {
91             for(int i = 0;i < M;i++)
92             {
93                  tmp[i] = in_tab[i];
94             }
95             FCTIIkrok(N1, in_tab);
96             // Szacownie błędu w metryce Czebyszewa
97             epsmd = 0.0;
98             for(int i = 0;i < M;i++)
99             {
100                 epso = fabs((in_tab[i] - 2.0*tmp[i])/(double)N1)/3.0;
101                 if (epso > epsmd)
102                 {
103                      epsmd = epso;
104                 }
105            }
106            if (epsmd < epsilon)
107            {
108                 break;
109            }
110            if (N1 == N)
111            {
112                 break;
113            }
114            FCTIIblok(N1, in_tab + N1);
115            N1 *= 2;
116       }
117       while(1);
118       // Wypisz odpowiednie informacje wg. punktu 8
119       // szybkiego algorytmu adaptacyjnego
120       delete[] tmp;
121  }
```

ISBN 978-83-7283-886-5