**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS**

**TAIKOMOSIOS MATEMATIKOS KATEDRA**

**Jonas Ribokas**

# DISKREČIOJO LE GALL SPEKTRO APSKAIČIAVIMO DVIMAČIO VAIZDO FRAGMENTAMS ALGORITMŲ ANALIZĖ

Magistro darbas

**Vadovas**
**prof. dr. J. Valantinas**

**KAUNAS, 2011**

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS**

**TAIKOMOSIOS MATEMATIKOS KATEDRA**

**TVIRTINU**
**Katedros vedėjas**
**doc. dr. N. Listopadskis**
**arba**
**prof. habil.dr. V.Pekarskas**
**2011 06 02**

# DISKREČIOJO LE GALL SPEKTRO APSKAIČIAVIMO DVIMAČIO VAIZDO FRAGMENTAMS ALGORITMŲ ANALIZĖ

Taikomosios matematikos magistro baigiamasis darbas

**Vadovas**
**(  ) prof. dr. J. Valantinas**
**2011 06 02**

**Recenzentas**
**(  ) doc. dr. R. Rindzevičius**
**2011 06 02**

**Atliko**
**FMMM 9 gr. stud.**
**(  ) J. Ribokas**
**2011 05 30**

**KAUNAS, 2011**

KVALIFIKACINĖ KOMISIJA

**Pirmininkas:** Leonas Saulis, profesorius (VGTU)

**Sekretorius:** Eimutis Valakevičius, docentas (KTU)

**Nariai**: Algimantas Jonas Aksomaitis, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, habil. dr., vyriausiasis analitikas (DnB NORD Bankas)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

# SANTRAUKA

Darbe analizuojama diskrečioji Le Gall transformacija (DLGT) yra viena iš bangelių transformacijų, kurios naudojamos skaitmeninių vaizdų glaudinime. Vaizdų glaudinimas ypač svarbus, kai kalbama apie vaizdų siuntimą mažo pralaidumo kompiuteriniu kanalu. Šiai problemai spręsti pasitelkiama diskrečioji Le Gall transformacija, bei didelis dėmesys darbe skirtas vaizdo fragmento DLGT spektrui. Šio spektro radimui sukurtas ir realizuotas naujas algoritmas − greitoji procedūra. Palyginamosios analizės metu įrodyta, kad sukurta greitoji procedūra randa vaizdo fragmento DLGT spektrą daugiau nei 50 kartų greičiau nei tiesiogiai taikant DLGT vaizdo fragmentui.

Darbo pradžioje aptariamos pagrindinės bangelių ir jų transformacijų savybės. Šių savybių svarba iliustruojama pateiktais pavyzdžiais ir sąsajomis su taikymu praktikoje. Viena iš svarbesnių aptariamų savybių būtų transformacijos lokalizavimas erdvėje. Nors DLGT pilnai netenkina šios savybės, tačiau darbe apžvelgtas dekoreliacijos metodas leidžia išspręsti šią problemą. Be to, pateikiami išsamūs DLGT algoritmai vienmačiams ir dvimačiams vaizdams. Šių algoritmų dėka buvo sukurta programinė įranga, padedanti analizuoti DLGT. Galiausiai darbo pabaigoje pateikiama sukurtos programinės įrangos instrukcija.

# SUMMARY

In this paper the main attention is dedicated to discrete Le Gall transformation (DLGT), the family of discrete wavelets transformations, which usage in the image compression is quite popular. Image compression is very important when it comes to low bandwidth network and computer channels. To deal with this problem the discrete Le Gall wavelets are brought up and focused on the image fragment DLGT spectrum. For calculation of this spectrum the new fast procedure is suggested and realized. In comparative analysis the fast procedure is proved to work more than 50 times faster than applying DLGT direct to image fragment.

In the beginning of this paper the main concepts of wavelets and their transformations are analyzed. An importance of these properties is illustrated by examples and applications in practice. One of the most important properties is localization in space. Even though DLGT does not fully meet this property decorrelation technique, introduced in paper, benefits in solving localization problem. Moreover, DLGT algorithms are proposed for one-dimensional and 2-dimensional images. According to these algorithms the new program was created, which is dedicated for image analysis. Finally, in the end of paper the instructions of program are presented.

TURINYS

Lentelių sąrašas ..... 7
Paveikslų sąrašas ..... 8
Įvadas ..... 9
1 Diskrečiosios bangelių transformacijos (DBT) ..... 9
1.1 Bendrosios bangelių savybės ..... 9
1.2 DBT realizavimo ypatumai ..... 11
1.3 Diskrečioji Le Gall transformacija (DLGT): savybės, skaičiavimo algoritmai. ..... 14
1.3.1 Bazinė DLGT vienmačiu ir daugiamačiu atvejais ..... 14
1.3.2 DLGT su daline blokų dekoreliacija ..... 20
2 Greitų DLGT spektro apskaičiavimo vaizdo fragmentams procedūrų sudarymas ir jų efektyvus tyrimas ..... 22
2.1 DLGT spektro vienmačio vaizdo fragmentams radimas ..... 22
2.2 DLGT spektro dvimačio vaizdo fragmentams nustatymas ..... 22
2.3 Palyginamosios eksperimentinės analizės rezultatai ..... 25
3 Programinė realizacija ir instrukcija vartotojui ..... 26
Išvados ..... 29
Literatūra ..... 30
Priedai ..... 31

# LENTELIŲ SĄRAŠAS

1.1 lentelė. Žemo ir aukšto dažnio filtrų koeficientai ..... 14
1.2 lentelė. Vienmačio signalo žymėjimas ir pavyzdys ..... 15
1.3 lentelė. Vienmačio signalo DLGT spektro radimas ..... 17
2.1 lentelė. Greičio testo rezultatai ..... 25
2.2 lentelė. Greičio išlošis ..... 25

# PAVEIKSLŲ SĄRAŠAS

1.1 pav. Skaitmeninio signalo samprata ..... 12
1.2 pav. Signalo spektro radimas ir atkūrimas naudojant DLGT ..... 12
1.3 pav. Praktinis DLGT pritaikymas mažo pralaidumo kanalui ..... 13
1.4 pav. Lokaliai progresyvus vaizdų kodavimas ..... 14
1.5 pav. Žemo dažnio filtro taikymo $i = 1$ iteracijai pavyzdys ..... 16
1.6 pav. Aukšto dažnio filtro taikymo $i = 1$ iteracijai pavyzdys ..... 16
1.7 pav. Žemo dažnio filtro taikymo pavyzdys, kai $n = 3$ ..... 16
1.8 pav. Aukšto dažnio filtro taikymo pavyzdys, kai $n = 3$ ..... 17
1.9 pav. DLGT transformacija pagal stulpelius dvimačiui signalui ..... 18
1.10 pav. DLGT transformacija pagal eilutes dvimačiui signalui ..... 18
1.11 pav. DLGT transformacija $N = 4$ dvimačiui signalui ..... 18
1.12 pav. DLGT transformacija $N = 8$ dvimačiui signalui ..... 19
1.13 pav. DLGT transformacija $N = 4$ dvimačiui signalui ..... 19
1.14 pav. Atvirkštinė DLGT transformacija $N = 4$ dvimačiui signalui ..... 19
1.15 pav. Atvirkštinė DLGT transformacija $N = 8$ dvimačiui signalui ..... 19
1.16 pav. DLGT transformacija su dekoreliacija, kai $m = 3$ ir $n = 4$ ..... 21
2.1 pav. Vaizdo fragmento spektro radimo schema ..... 23
2.2 pav. Vaizdo fragmento spektro radimo greitoji procedūra ..... 24
3.1 pav. Pagrindinis programos langas ..... 26
3.2 pav. Sudėtingojo greičio testo valdymo langas ..... 28
3.3 pav. 400*400 pikseliu vaizdo ir spektro pavyzdys ..... 28

# ĮVADAS

Viena iš daugelio skaitmeninių vaizdų tyrimo krypčių yra vaizdų kodavimo (suglaudimo) technologijos, kurių veikimas grindžiamas diskrečiosiomis bangelių transformacijomis. Šiame darbe bus aptartos pagrindinės diskrečiųjų bangelių transformacijų savybės, jų skaičiavimo algoritmai, ir pagrindinis dėmesys bus skirtas diskrečiajai Le Gall transformacijai (DLGT). Kodėl DLGT svarbi? Todėl, kad šios transformacijos savybės (lokalizacija pagal dažnį ir erdvėje, greitos skaičiavimo procedūros) leidžia DLGT pritaikyti įvairiems praktiniams tikslams. Vienas tokių tikslų − vaizdų suglaudimas ir jų (ypač didelės raiškos) perdavimas mažo pralaidumo kompiuteriniu kanalu. Šiai problemai spręsti naudojamas lokaliai progresyvus vaizdų kodavimas, realizuojamas DLGT ar kitų bangelių transformacijų bazėje. Kadangi lokaliai progresyvaus vaizdų kodavimui argumentuoti būtina, kad naudojama bangelių transformacija būtų pilnai lokalizuota erdvėje (bent jau aukštų dažnių srityje).

Darbo eigoje buvo sukurtas algoritmas (greitoji procedūra) skirtas pasirinktų fragmentų DLGT spektro radimui, remiantis apdorojamo vaizdo DLGT spektru. Buvo atlikta išsami palyginamoji sudarytos procedūros efektyvumo (greičio požiūriu) analizė. Taip pat buvo sukurta minėtai analizei skirta programinė įranga.

# 1 DISKREČIOSIOS BANGELIŲ TRANSFORMACIJOS (DBT)

## 1.1 BENDROSIOS BANGELIŲ SAVYBĖS

Pirmiausiai apibrėžkime tolydžią bangelių transformaciją

$$\gamma(s,\tau) = \int f(t)\psi^*_{s,\tau}(t)\,dt \tag{1.1}$$

kitaip tariant, tai yra funkcijos $f(t)$ sukomponavimas į aibę bazinių funkcijų $\psi_{s,\tau}(t)$, vadinamų bangelėmis. Čia funkcija $\gamma(s,\tau)$ vadinama spektru, o $s$ ir $\tau$ yra mastelio ir postūmio parametrai. Bangelių transformacija turi turėti atvirkštine transformaciją

$$f(t) = \iint \gamma(s,\tau)\psi_{s,\tau}(t)\,d\tau ds\,. \tag{1.2}$$

Bangele vadiname tokia funkcija $\psi_{s,\tau}(t)$

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right), \tag{1.3}$$

čia $\psi(t)$ bazinė bangelės funkcija. $\frac{1}{\sqrt{s}}$ yra skirta energijos normalizavimui skirtingiems masteliams. Svarbu pabrėžti, kad bangelės funkcija nėra griežtai apibrėžta (priešingai nei Furje transformacija). Tačiau bangelės funkcija turi tenkinti priimtinumo ir reguliarumo savybes.

Bangelės funkcija yra priimtina, jei jos vidurkis yra lygus 0, t.y.

$$\int_{-\infty}^{\infty} \psi(t)dt = 0\,. \tag{1.4}$$

Tai reiškia, kad bangelės funkcija turi banguoti. Jei apibrėžtume $\Psi(\omega)$ kaip bangelės funkcijos $\psi(t)$ Furje transformaciją, tai priimtinumo savybe galima užrašyti taip

$$\int \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < +\infty\,. \tag{1.5}$$

Tai garantuoja, kad tiesioginė ir atvirkštinė transformacijos atliekamos be informacijos praradimo.

Bangelės funkcijos reguliarumo savybė reiškia, kad bangelės funkcija turi pakankamai greitai slopti. Išskleiskime (1.1) transformaciją Teiloro eilute taške $t = 0$ (paprastumo dėlei postūmio parametras $\tau = 0$)

$$\gamma(s,0) = \frac{1}{\sqrt{s}}\left[\sum_{p=0}^{n} f^{(p)}(0)\int \frac{t^p}{p!}\psi\left(\frac{t}{s}\right)dt + O(n+1)\right], \tag{1.6}$$

čia $f^{(p)}(t)$ funkcijos $f(t)$ p-toji išvestinė. Pažymėkime $M_p$ − bangelės momentą

$$M_p = \int t^p \psi(t)\,dt \tag{1.7}$$

tada (1.6) formulę galima perrašyti

$$\gamma(s,0) = \frac{1}{\sqrt{s}}\left[\begin{array}{l} f(0)M_0 s + \frac{f^{(1)}(0)}{1!}M_1 s^2 + \frac{f^{(2)}(0)}{2!}M_2 s^3 + \ldots \\ + \frac{f^{(n)}(0)}{n!}M_n s^{n+1} + O\left(s^{n+2}\right) \end{array}\right]. \tag{1.8}$$

Iš priimtinumo savybės žinome, kad pirmasis momentas lygus 0. Esant glodžiam signalui $f(t)$, jeigu pavyktų ir kitus momentus iki $M_n$ priartinti prie 0, tai spektro $\gamma(s,\tau)$ koeficientai sloptų tokiu greičiu kaip $s^{n+2}$. Ši savybė vadinama momentų slopimu arba aproksimavimo eilė. Taigi reguliarumo savybe vadinsime bangelės greitu slopimu.

Išanalizavus formalų tolydžios bangelių transformacijos aprašymą galima teigti, kad svarbiausios savybės yra atvirkštinės transformacijos egzistavimas, bei gebėjimas be nuostolių atkurti pradinę funkciją/signalą. Pati bangelės funkcija nėra griežtai apibrėžta, tačiau turi tenkinti jau aptartas priimtinumo ir reguliarumo savybes.

**Diskrečiosios bangelės**

Praktikoje tolydžioji bangelių transformacija sunkiai įgyvendinama, kadangi kompiuterinė technika dirba tik su baigtiniais skaičiais. Norint panaudoti bangelių savybes praktikoje, tenka išvesti taip vadinamas diskrečiąsias bangeles, kurių transformacijos būtų nesunkiai įgyvendinamos kompiuteriniuose įrenginiuose.

Viena iš pagrindinių tolydžių bangelių transformacijos trūkumų yra pertekliškumas. Transformacijai yra naudojamos tolydžioji mastelio ir tolydžioji postūmio funkcijos. Šios funkcijos toli gražu nesudarys ortogonalios bazės (1.4), todėl gauti spektro koeficientai bus pertekliniai. Šiai problemai spręsti įvedama diskrečiųjų bangelių sąvoka.

Diskrečiųjų bangelių funkcija pastumta bei pakeistas jos mastelis yra per fiksuotą žingsnių skaičių. Todėl (1.3) formulę pakeičiame

$$\psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}} \psi\left( \frac{t - k\tau_0 s_0^j}{s_0^j} \right), \tag{1.9}$$

čia $j$ ir $k$ yra sveikieji skaičiai, $s_0 > 1$ išsiplėtimo dydis, $\tau_0$ – postūmio parametras. Kitaip tariant, diskrečiąja bangele vadinsime tolydžios bangelių funkcijos diskrečiais laiko momentais fiksuotas reikšmes.

Dar viena iš diskrečiųjų bangelių savybių yra ortogonalumas

$$\int \psi_{j,k}(t) \psi^*_{m,n}(t)\, dt = \begin{cases} c, \text{ kai } j = m \text{ ir } k = n \\ 0 \text{ kitur} \end{cases}. \tag{1.10}$$

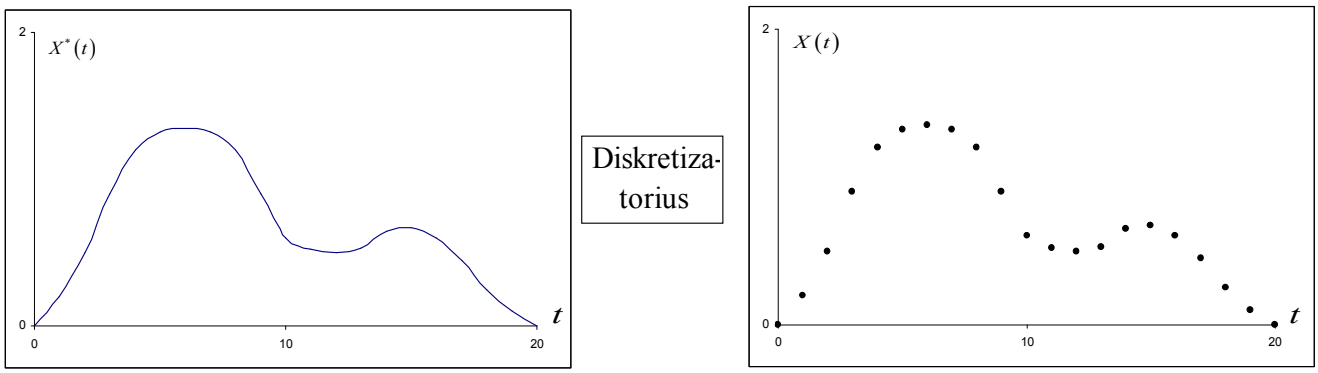
Kai $c = 1$, tai funkcija $\psi_{j,k}(t)$ vadinama ortonormuota.

Atvirkštinė diskrečioji bangelių transformacija gaunama

$$f(t) = \sum_{j,k} \gamma(j,k) \psi_{j,k}(t) \tag{1.11}$$

## 1.2 DBT REALIZAVIMO YPATUMAI

Skaitmeninis signalas gaunamas fiksuojant tolydųjį signalą vienodais laiko tarpais (1.1 paveikslas). Suformuojamas duomenų vektorius $X = \left( X(0) X(1) \ldots X(N-1) \right)^T$, kurį vadinsime $N$ dydžio skaitmeniniu signalu arba vienmačiu skaitmeniniu vaizdu.




**1.1 pav. Skaitmeninio signalo samprata**

Galima užrašyti diskrečiąją bangelių transformaciją (DBT) matricine forma

$$Y_X = \left(Y(0)Y(1)\ldots Y(N-1)\right)^T = \frac{1}{N}T(n)\cdot X, \tag{1.12}$$

čia $T(n)-\ N\times N$ dydžio transformacijos matrica. $Y_x$ vadinamas spektru. Kaip buvo minėta, DBT tenkina ortogonalumo savybę, t.y.

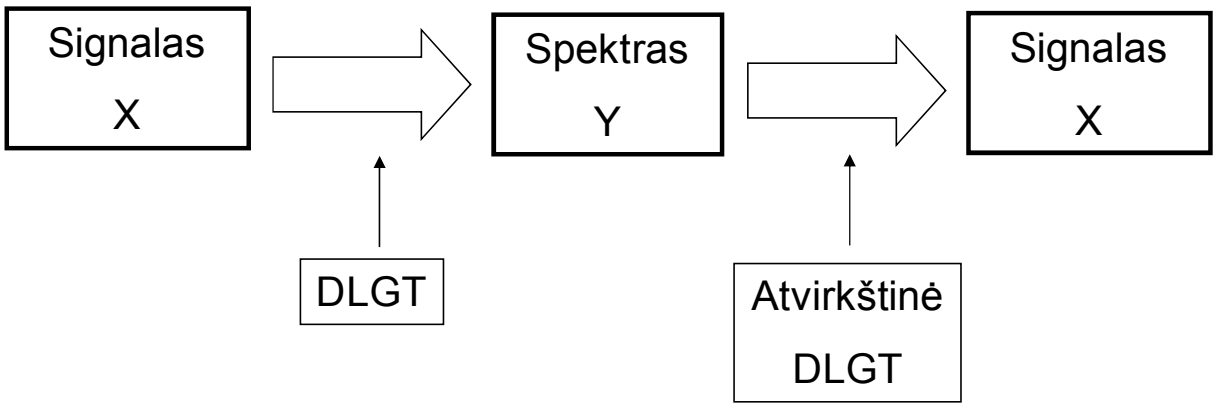
$$T^T(n)T(n) = N\cdot E(n), \tag{1.13}$$

kai $E(n)$ yra vienetinė matrica. Remdamiesi šia savybe galime vienareikšmiškai apibrėžti atvirkštinę DBT

$$X = T^T(n)Y_X \tag{1.14}$$

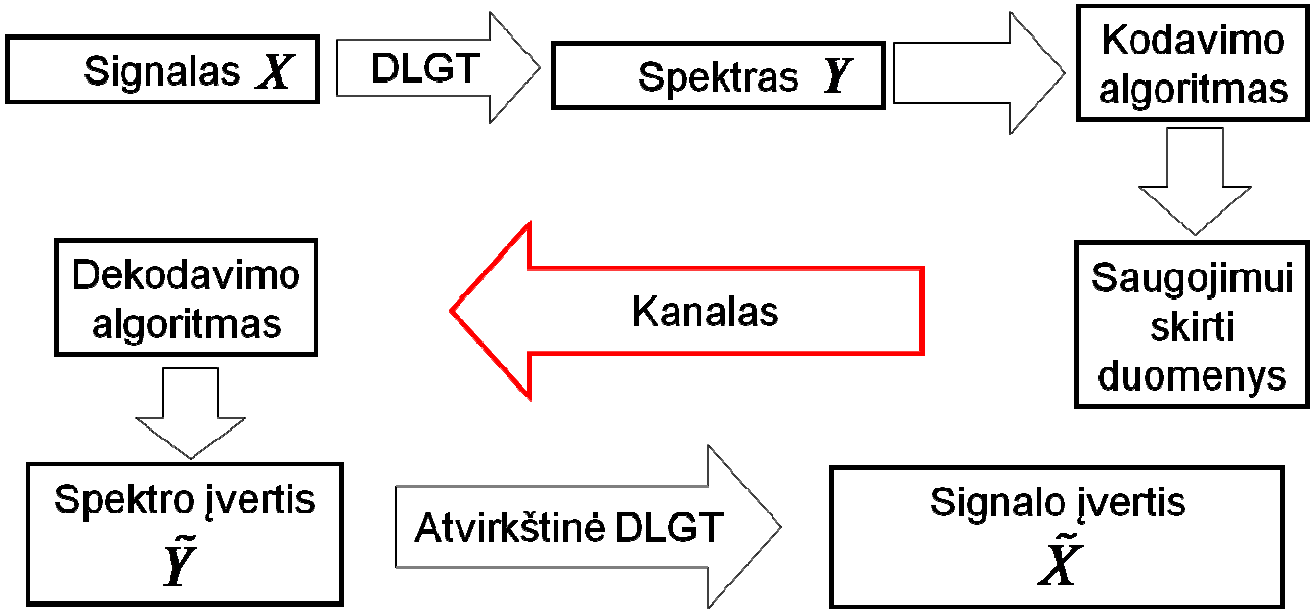
Egzistuoja keletas DBT matricos $T(n)$ formų. Vienos iš žymesnių Haar, Le Gall bei Daubechy sukurtos diskrečiosios bangelių transformacijos. Teorijoje dažnai sutinkama Haar DBT. Šiame darbe visą dėmesį skirsime Le Gall diskrečiajai bangelių transformacijai − DLGT. Kaip apibrėžiama DLGT matrica, nagrinėsime kitame skyriuje.

Viena iš pagrindinių diskrečiųjų bangelių transformacijos savybių yra gebėjimas ne tik rasti signalo spektrą, bet ir turint spektrą vienareikšmiškai atkurti pradinį signalą



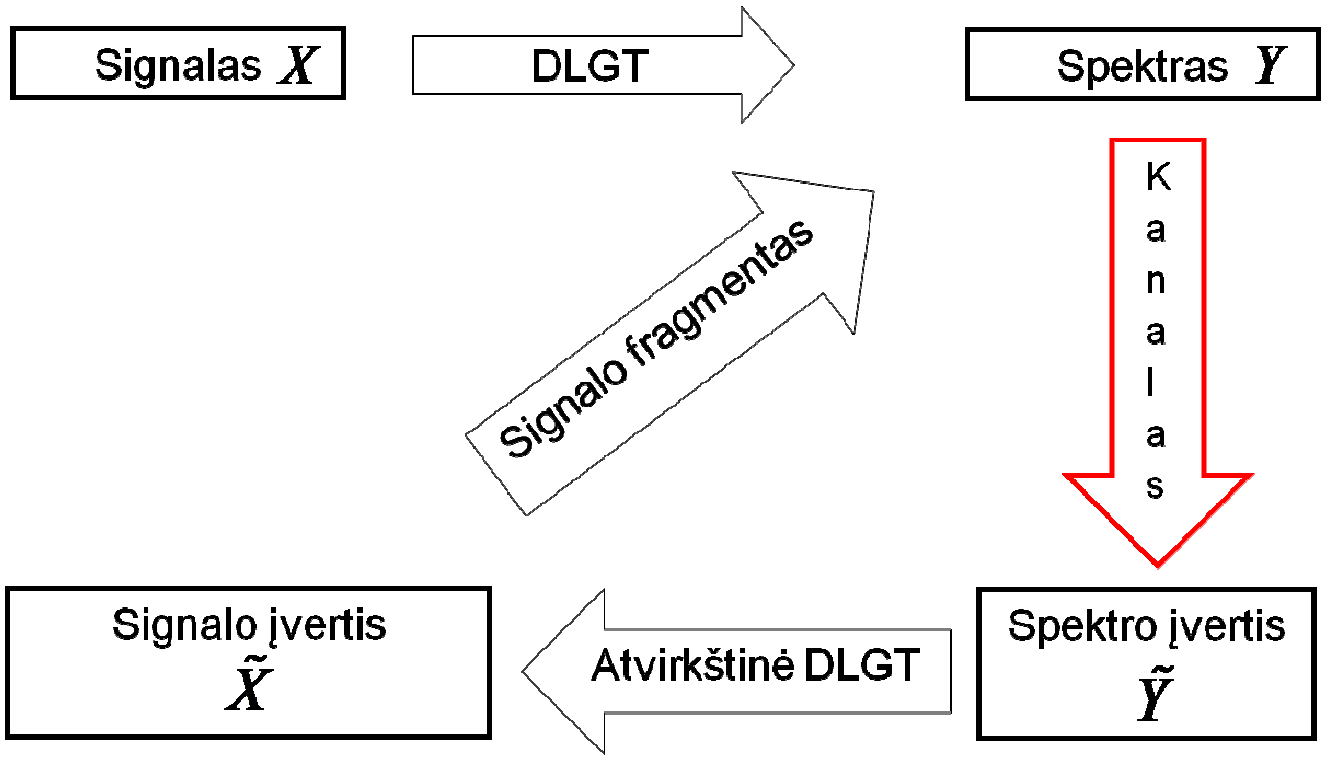

**1.2 pav. Signalo spektro radimas ir atkūrimas naudojant DLGT**

Praktikoje pasitaiko problemos, kurios susijusios su mažo pralaidumo kompiuteriniais kanalais. Būtent šioje vietoje ir praverčia lokaliai progresyvus vaizdų kodavimas, kurį įgyvendinti padeda DLGT.




**1.3 pav. Praktinis DLGT pritaikymas mažo pralaidumo kanalui**

1.3 paveiksle matomas praktinis DLGT pritaikymas. Pirmiausiai skaitmeniniam signalui taikoma DLGT ir gaunamas spektras. Vėliau kodavimo algoritmai (SPIHT, EZW) pasitelkdami ypatingą spektro struktūrą, stipriai suspaudžia spektrą ir gautus duomenis siunčia kompiuteriniu kanalu. Kai duomenys perduodami į kitą kanalo pusę, įsijungia dekodavimo algoritmai. Iš siuntimui skirtų duomenų (priklausomai nuo nustatyto suspaudimo lygio) gaunamas spektro įvertis. Tada taikoma atvirkštinė DLGT ir gaunamas grubus signalo/vaizdo įvertis. Jeigu vartotoją tenkina tokia signalo kokybė, tai darbas baigtas. Tačiau vartotojas pagal pageidavimą gali išryškinti tam tikrą vaizdo fragmentą. Tai pavaizduoja 1.4 paveiksle esanti supaprastinta schema. Kadangi DLGT (ir kitų DBT) spektro koeficientai yra susieti su tam tikrais vaizdo fragmentais, tai vartotojui pateikus užklausą spektre taikant greitąją procedūrą išrenkami koeficientai ir siunčiami kanalu. Tokiu būdu papildomas informacijos kiekis išryškina tik vaizdo fragmentą. Kaip alternatyva greitajai procedūrai, gali būti tiesioginis DLGT spektro taikymas norimam vaizdo fragmentui. Šio darbo tikslas yra palyginti abu būdus norimam vaizdo fragmentui išryškinti.




**1.4 pav. Lokaliai progresyvus vaizdų kodavimas**

## 1.3 DISKREČIOJI LE GALL TRANSFORMACIJA (DLGT): SAVYBĖS, SKAIČIAVIMO ALGORITMAI.

### 1.3.1 BAZINĖ DLGT VIENMAČIU IR DAUGIAMAČIU ATVEJAIS

**Vienmatis atvejis**

Diskrečioji Le Gall transformacija (DLGT) turi 5 žemo dažnio ir 3 aukšto dažnio filtrų koeficientus.

**1.1 lentelė**

**Žemo ir aukšto dažnio filtrų koeficientai**

| | | |
|---|---|---|
| Žemo dažnio filtras | $h_0$ | - 1/8 |
| | $h_1$ | 1/4 |
| | $h_2$ | 3/4 |
| | $h_3$ | 1/4 |
| | $h_4$ | - 1/8 |
| Aukšto dažnio filtras | $g_0$ | - 1/2 |
| | $g_1$ | 1 |
| | $g_2$ | - 1/2 |

Pažymėkime $N = 2^n$ ($n \in \square$) dydžio vienmatį signalą

$$X = \left(x_0\, x_1 \ldots x_{N-1}\right)^T = \left(o_0^{(0)}\, e_0^{(0)}\, o_1^{(0)}\, e_1^{(0)}\ \ldots o_{N/2-1}^{(0)}\, e_{N/2-1}^{(0)}\right)^T, \tag{1.15}$$

čia $o_k^{(0)}$ žymimos lyginės signalo reikšmės, o $e_k^{(0)}$ - nelyginės.

**1.2 lentelė**

**Vienmačio signalo žymėjimas ir pavyzdys**

| | | | Pavyzdys |
|---|---|---|---|
| Signalas | $x_0$ | $o_0$ | 1 |
| | $x_1$ | $e_0$ | 2 |
| | $x_2$ | $o_1$ | 3 |
| | $x_3$ | $e_1$ | 4 |
| | $x_4$ | $o_2$ | 5 |
| | $x_5$ | $e_2$ | 6 |
| | $x_6$ | $o_3$ | 7 |
| | $x_7$ | $e_3$ | 8 |

Signalo $X$ DLGT spektras $Y$ gaunamas per $n$ iteracijų. Apibrėžkime tarpinių duomenų poaibius, kuriuos gauname taikydami žemo ir aukšto dažnio filtrus. Taikydami žemo dažnio filtrą gauname vektorių

$$S^{(i)} = \left(s_0^{(i)}\, s_1^{(i)}\, s_2^{(i)} \ldots s_{2^{n-i}-1}^{(i)}\right)^T = \left(o_0^{(i)}\, e_0^{(i)}\, o_1^{(i)}\, e_1^{(i)}\ \ldots o_{2^{n-i-1}-1}^{(i)}\, e_{2^{n-i-1}-1}^{(i)}\right)^T \tag{1.16}$$

ir taikydami aukšto dažnio filtrą gauname

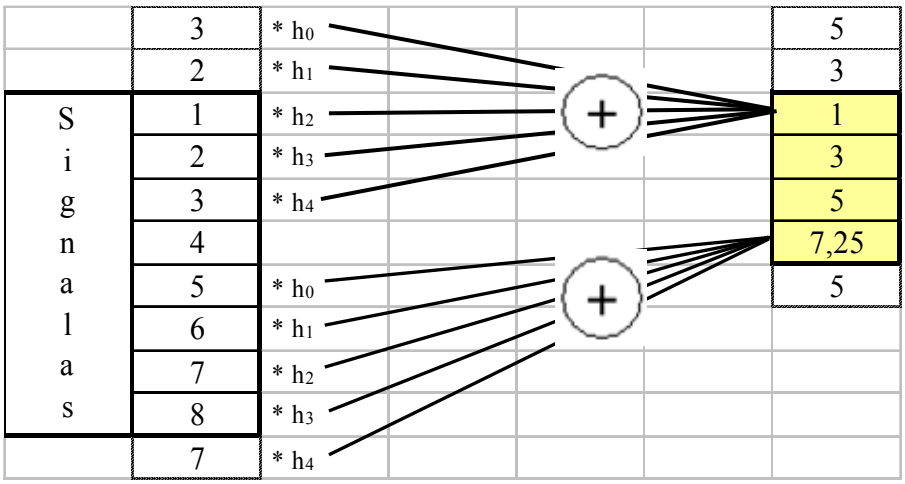
$$D^{(i)} = \left(d_0^{(i)}\, d_1^{(i)}\, d_2^{(i)} \ldots d_{2^{n-i}-1}^{(i)}\right)^T, \tag{1.17}$$

čia $i \in \{1,2,\ldots,n\}$, bei $S^{(0)} = X$.

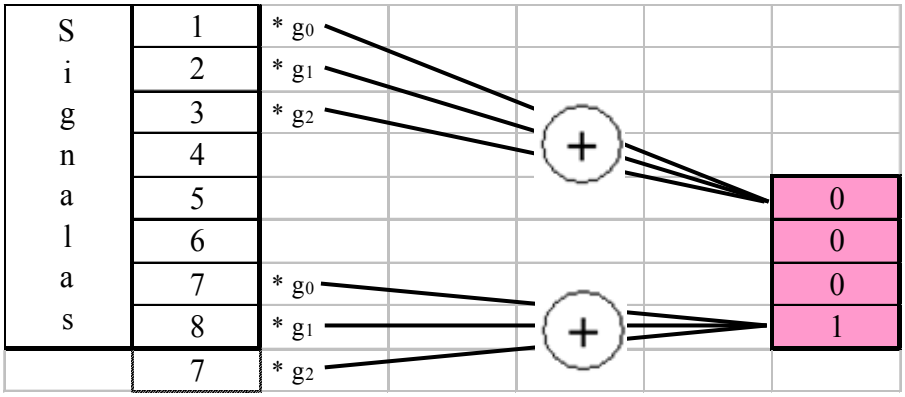
Norėdami paskaičiuoti tarpinių duomenų vektorius, kiekvienam vektoriaus nariui taikome formules (aukšto ir žemo dažnio filtrus):

$$\begin{aligned} d_k^{(i)} &= e_k^{(i-1)} - \frac{1}{2}\left(o_k^{(i-1)} + o_{k+1}^{(i-1)}\right), \\ s_k^{(i)} &= o_k^{(i-1)} + \frac{1}{4}\left(d_{k-1}^{(i)} + d_k^{(i)}\right), \end{aligned} \tag{1.18}$$

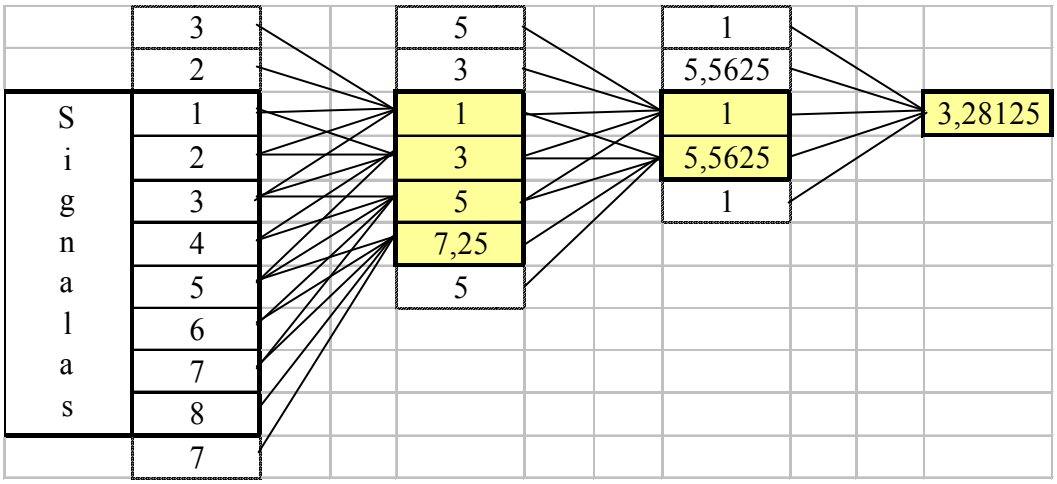
visiems $k = 0,1,\ldots,2^{n-i}-1$, kur $o_{2^{n-1}}^{(i-1)} := o_{2^{n-1}-1}^{(i-1)}$, $d_{-1}^{(i)} := d_0^{(i)}$ ir $i \in \{1,2,\ldots,n\}$.



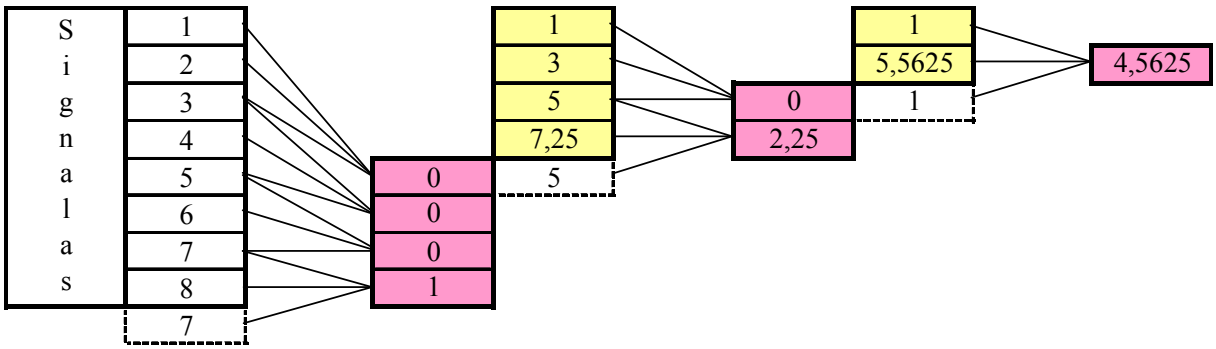

**1.5 pav. Žemo dažnio filtro taikymo $i = 1$ iteracijai pavyzdys**




**1.6 pav. Aukšto dažnio filtro taikymo $i = 1$ iteracijai pavyzdys**




**1.7 pav. Žemo dažnio filtro taikymo pavyzdys, kai $n = 3$**




**1.8 pav. Aukšto dažnio filtro taikymo pavyzdys, kai** $n = 3$

Suradus tarpinius duomenų vektorius, galime apibrėžti signalo $X$ DLGT spektrą $Y$:

$$Y = \left( s_0^{(n)}\, d_0^{(n)}\, d_0^{(n-1)}\, d_1^{(n-1)}\, d_0^{(n-2)} d_1^{(n-2)} d_2^{(n-2)} d_3^{(n-2)} \ldots d_0^{(1)}\, d_1^{(1)} \ldots d_{2^{n-1}-1}^{(1)} \right)^T . \tag{1.19}$$

Norint paskaičiuoti DLGT spektrą signalui, susiduriame su „krašto problema“. Tai yra viena iš problemų, kodėl DLGT yra tik dalinai lokalizuota erdvėje. Kaip matome, šia problemą išsprendėme laikydami, kad signalo galai buvo lyg veidrodžio atspindys, taip pratęsiant signalo kraštus.

**1.3 lentelė**

**Vienmačio signalo DLGT spektro radimas**

| Iteracijos | i = | 0 | | | 1 | | | 2 | | | 3 | | | Spektras |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Signalas | $x_0$ | $o_0$ | 1 | $s_0$ | $o_0$ | 1 | $s_0$ | $o_0$ | 1 | $s_0$ | 3,28125 | 3,28125 | 3,28125 |
| | | $x_1$ | $e_0$ | 2 | $s_1$ | $e_0$ | 3 | $s_1$ | $e_0$ | 5,5625 | $d_0$ | 4,5625 | 4,5625 | 4,5625 |
| | | $x_2$ | $o_1$ | 3 | $s_2$ | $o_1$ | 5 | $d_0$ | | 0 | | | | 0 |
| | | $x_3$ | $e_1$ | 4 | $s_3$ | $e_1$ | 7,25 | $d_1$ | | 2,25 | | | | 2,25 |
| | | $x_4$ | $o_2$ | 5 | $d_0$ | | 0 | | | | | | | 0 |
| | | $x_5$ | $e_2$ | 6 | $d_1$ | | 0 | | | | | | | 0 |
| | | $x_6$ | $o_3$ | 7 | $d_2$ | | 0 | | | | | | | 0 |
| | | $x_7$ | $e_3$ | 8 | $d_3$ | | 1 | | | | | | | 1 |

**Dvimačiu atveju**

Tarkime, kad turime dvimatį signalą $X$, kurio dydis $N \times N = 2^n \times 2^n$, $(n \in \square)$. Norint rasti šio signalo DLGT spektrą $Y$, pirmiausia reikėtų suskaidyti signalą $X$ į vienmačius vektorius pagal stulpelius ir jiems pritaikyti DLGT transformaciją vienmačiui signalui.

| DLGT | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

**1.9 pav. DLGT transformacija pagal stulpelius dvimačiui signalui**

Tada gautą tarpinę duomenų matricą suskaidome pagal eilutes ir joms panaudojame DLGT transformaciją vienmačiam signalui.

| DLGT | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| → | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| → | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 |
| → | x3 | x4 | x5 | x6 | x7 | x8 | x9 | x10 |
| → | x4 | x5 | x6 | x7 | x8 | x9 | x10 | x11 |
| → | x5 | x6 | x7 | x8 | x9 | x10 | x11 | x12 |
| → | x6 | x7 | x8 | x9 | x10 | x11 | x12 | x13 |
| → | x7 | x8 | x9 | x10 | x11 | x12 | x13 | x14 |
| → | x8 | x9 | x10 | x11 | x12 | x13 | x14 | x15 |

**1.10 pav. DLGT transformacija pagal eilutes dvimačiui signalui**

Atlikus šį žingsnį, gauname dvimačio signalo $X$ DLGT spektrą $Y$.

Pateiksime keletą pavyzdžių, kaip veikia DLGT transformacijos algoritmas. Paprastumo dėlei algoritmas buvo realizuotas programiškai. Panaudoti signalai yra nedideli ir nesudėtingi, tačiau algoritmas puikiai susidorotų ir su žymiai didesniais signalais. Šis algoritmas veikia su realiųjų skaičių signalais.

| N = | 4 | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 2 | 3 | 4 | 5 |
| 3 | 4 | 5 | 6 |
| 4 | 5 | 6 | 7 |

DLGT

| N = | 4 | | |
|---|---|---|---|
| 3,25 | 2,25 | 0 | 1 |
| 2,25 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |

**1.11 pav. DLGT transformacija $N = 4$ dvimačiui signalui**

| N = | 8 | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

DLGT

| N = | 8 | | | | | | |
|---|---|---|---|---|---|---|---|
| 5,5625 | 4,5625 | 0 | 2,25 | 0 | 0 | 0 | 1 |
| 4,5625 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2,25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**1.12 pav. DLGT transformacija $N = 8$ dvimačiui signalui**

| N = | 4 | | |
|---|---|---|---|
| 1,1 | 2,2 | 3,2 | 4,4 |
| 2,2 | 3,3 | 4,3 | 5,5 |
| 3,3 | 4,4 | 5,5 | 6,6 |
| 4,4 | 5,5 | 6,6 | 7,7 |

DLGT

| N = | 4 | | |
|---|---|---|---|
| 3,575 | 2,475 | 0 | 1,1 |
| 2,475 | 4,44E-16 | -8,88E-16 | -8,88E-16 |
| -1,67E-16 | 1,11E-16 | -4,44E-16 | 0 |
| 1,1 | -6,66E-16 | -4,44E-16 | 8,88E-16 |

**1.13 pav. DLGT transformacija $N = 4$ dvimačiui signalui**

1.12 pav. pateiktame DLGT transformacijos pavyzdyje galime pastebėti, šalutinį kompiuterio skaičių apvalinimo poveikį, kuris praktiškai neturi įtakos algoritmo tikslumui.

Patikrinsime algoritmo teisingumą, panaudodami gautuosius spektrus, kaip duomenis atvirkštinei DLGT transformacijai. Signalas $X$ ir spektras $Y$ turi tenkinti $X \Rightarrow Y \Rightarrow X$ savybę.

| N = | 4 | | |
|---|---|---|---|
| 3,575 | 2,475 | 0 | 1,1 |
| 2,475 | 4,44E-16 | -8,88E-16 | -8,88E-16 |
| -1,67E-16 | 1,11E-16 | -4,44E-16 | 0 |
| 1,1 | -6,66E-16 | -4,44E-16 | 8,88E-16 |

Inverse DLGT

| N = | 4 | | |
|---|---|---|---|
| 1,10000000000003 | 2,2 | 3,29999999999999 | 4,4 |
| 2,2 | 3,3 | 4,39999999999999 | 5,5 |
| 3,3 | 4,4 | 5,5 | 6,6 |
| 4,4 | 5,5 | 6,6 | 7,7 |

**1.14 pav. Atvirkštinė DLGT transformacija $N = 4$ dvimačiui signalui**

Kaip ir buvo galima tikėtis, menkos spektro skaičiavimo paklaidos nežymiai pablogino pradinio spektro radimą.

| N = | 8 | | | | | | |
|---|---|---|---|---|---|---|---|
| 5,5625 | 4,5625 | 0 | 2,25 | 0 | 0 | 0 | 1 |
| 4,5625 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2,25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Inverse DLGT

| N = | 8 | | | | | | |
|---|---|---|---|---|---|---|---|
| 1.0 | 2.0 | 3.0 | 4.0 | 5.0 | 6.0 | 7.0 | 8.0 |
| 2.0 | 3.0 | 4.0 | 5.0 | 6.0 | 7.0 | 8.0 | 9.0 |
| 3.0 | 4.0 | 5.0 | 6.0 | 7.0 | 8.0 | 9.0 | 10.0 |
| 4.0 | 5.0 | 6.0 | 7.0 | 8.0 | 9.0 | 10.0 | 11.0 |
| 5.0 | 6.0 | 7.0 | 8.0 | 9.0 | 10.0 | 11.0 | 12.0 |
| 6.0 | 7.0 | 8.0 | 9.0 | 10.0 | 11.0 | 12.0 | 13.0 |
| 7.0 | 8.0 | 9.0 | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 |
| 8.0 | 9.0 | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 | 15.0 |

**1.15 pav. Atvirkštinė DLGT transformacija $N = 8$ dvimačiui signalui**

Atlikę atvirkštinę DLGT transformaciją, gavome pradinį signalą, o tai reiškia, kad DLGT ir atvirkštinės DLGT transformacijos algoritmai teisingai atlieka savo darbą.

## 1.3.2 DLGT SU DALINE BLOKŲ DEKORELIACIJA

Jei mes perkelsime „krašto problemą“ nepersidengiantiems $2^m$ $\left(m \in \{1,2,\ldots,n-1\}\right)$ dydžio blokams, kurie sudaro signalą $X$, tada galėsime dekoreliuoti DLGT spektre $Y$ signalo blokus ne mažesnius nei $2^m$. Tada galėtume susieti šiuos blokus su tam tikrais spektro $Y$ koeficientais. Tokiu atveju naudosime modifikuotus aukšto ir žemo dažnio filtrus:

$$d_k^{(i)} = \begin{cases} e_k^{(i-1)} - o_k^{(i-1)}, \text{ kai } k \in \left\{\alpha_i \cdot t - 1 \,\middle|\, t = 1,2,\ldots,\beta_i\right\}, \\ e_k^{(i-1)} - \dfrac{1}{2}\left(o_k^{(i-1)} + o_{k+1}^{(i-1)}\right), \text{ kitu atveju}, \end{cases} \tag{1.20}$$

$$s_k^{(i)} = \begin{cases} o_k^{(i-1)} + \dfrac{1}{2} d_k^{(i)}, \text{ kai } k \in \left\{\alpha_i \cdot (t-1) \,\middle|\, t = 1,2,\ldots,\beta_i\right\}, \\ o_k^{(i-1)} + \dfrac{1}{4}\left(d_{k-1}^{(i)} + d_k^{(i)}\right), \text{ kitu atveju}, \end{cases} \tag{1.21}$$

visiems $k = 0,1,\ldots,2^{n-i}-1$, kur $i \in \{1,2,\ldots,n\}$. Čia:

$$\alpha_i = \begin{cases} 2^{m-i}, \text{ kai } i = 1,2,\ldots,m, \\ 1, \text{ kai } i = m+1, m+2,\ldots,n, \end{cases} \tag{1.22}$$

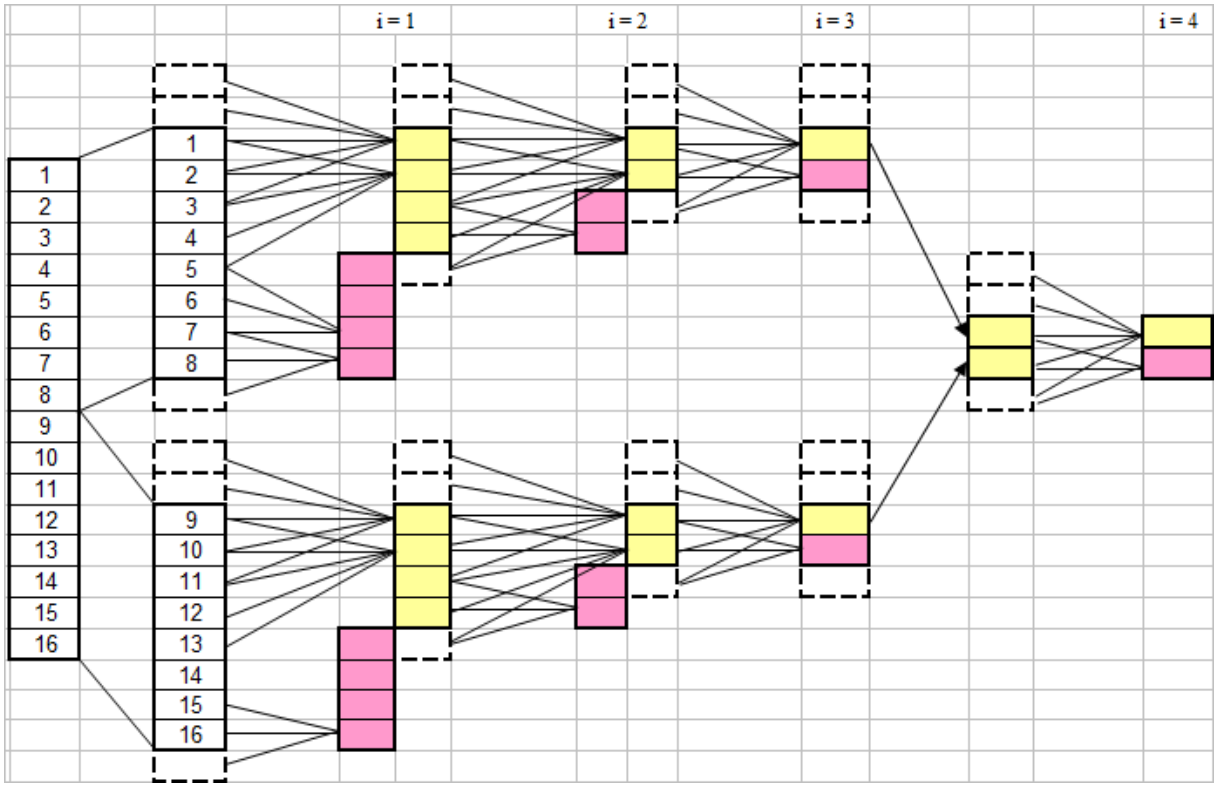
$$\beta_i = \begin{cases} 2^{n-m}, \text{ kai } i = 1,2,\ldots,m, \\ 2^{n-i}, \text{ kai } i = m+1, m+2,\ldots,n. \end{cases} \tag{1.23}$$

Apibrėžkime atvirkštinę DLGT

$$o_k^{(i-1)} = \begin{cases} s_k^{(i)} - \dfrac{1}{2} d_k^{(i)}, \text{ kai } k \in \left\{\alpha_i \cdot (t-1) \,\middle|\, t = 1,2,\ldots,\beta_i\right\}, \\ s_k^{(i)} - \dfrac{1}{4}\left(d_{k-1}^{(i)} + d_k^{(i)}\right), \text{ kitu atveju}, \end{cases} \tag{1.24}$$

$$e_k^{(i-1)} = \begin{cases} d_k^{(i)} + o_k^{(i-1)}, \text{ kai } k \in \left\{\alpha_i \cdot t - 1 \,\middle|\, t = 1,2,\ldots,\beta_i\right\}, \\ d_k^{(i)} + \dfrac{1}{2}\left(o_k^{(i-1)} + o_{k+1}^{(i-1)}\right), \text{ kitu atveju}, \end{cases} \tag{1.25}$$

visiems $k = 0,1,\ldots,2^{n-i}-1$, kur $i \in \{1,2,\ldots,n\}$.




**1.16 pav. DLGT transformacija su dekoreliacija, kai $m=3$ ir $n=4$**

Dekoreliaciją vaizdžiai paaiškina 1.15 paveiksle esanti schema. Tarkime, kad signalo dydis yra $N=16=2^4$, kai $n=4$. Tokiu atveju dekoreliacijos laipsnis gali būti $m\in\{1,2,3\}$. Tarkime, kad pasirinkome $m=3$, t.y. galėsime dekoreliuoti nemažesnius signalo blokus nei $2^3=8$. Tada 9 paveiksle matome, kad iki $i=3$ iteracijos, signalo $2^3=8$ dydžio blokams taikoma DLGT atskirai, ir tik $i=4$ iteracijoje žemo dažnio koeficientai sujungiami ir jiems pritaikoma DLGT, bei gaunamas galutinis spektras. Reikėtų atkreipti dėmesį, kad $i=3$ iteracijoje esančius žemo dažnio koeficientus galime dekoreliuoti, t.y. susieti su nepersidengiančiais signalo blokais. Šis pavyzdys parodo, kad dekoreliacijos pagalba galima DLGT padaryti pilnai lokalizuotą erdvėje − susieti spektro koeficientus su nepersidengiančiais signalo blokais.

Dekoreliacijos laipsnis $m$ nusako mažiausia dekoreliuojamo bloko dydi. Teoriškai yra siektina, kad $m$ būtų kuo mažesnis. Tai reiškia, kad tuo mažesnius signalo blokus (praktikoje− vaizdo fragmentus) galima būtų atkurti iš DLGT spektro. Tačiau šis apribojimai praktikoje neturi didelės įtakos. Jei kalbėtume apie vaizdų glaudinimą ir siuntimą mažo pralaidumo kanalu, mažesni nei $8\times8$ dydžio vaizdo fragmentai yra per maži, kad juose būtų įžvelgtos detalės. Todėl $m<3$ naudojimas praktikoje nėra tikslingas.

Atsižvelgiant į dekoreliacijos laipsnį, DLGT su daline blokų dekoreliacija yra pilnai lokalizuota erdvėje. Todėl ją galima panaudoti praktikoje − lokaliai progresyviame vaizdų kodavime.

# 2 GREITŲ DLGT SPEKTRO APSKAIČIAVIMO VAIZDO FRAGMENTAMS PROCEDŪRŲ SUDARYMAS IR JŲ EFEKTYVUS TYRIMAS

## 2.1 DLGT SPEKTRO VIENMAČIO VAIZDO FRAGMENTAMS RADIMAS

Norint rasti vienmačio vaizdo $X_j^{(i)}$ DLGT spektrą $Y_j^{(i)}$, kur $i \in \{m, m+1, \ldots, n-1\}$ ir $j \in \{0,1,\ldots,2^{n-i}-1\}$, galima taikyti greitąją procedūrą:

1. Pirmasis spektro koeficientas apskaičiuojamas pagal formulę

$$s_j^{(i)} = s_0^{(n)} - \frac{1}{2}\sum_{r=1}^{n-i}(-1)^{j_{r-1}} \cdot d_{j_r}^{(i+r)}, \tag{2.1}$$

$$j_0 = j,\ j_r = \left\lfloor \frac{j_{r-1}}{2} \right\rfloor,\ r = 1,2,\ldots,n-i.$$

2. Visi kiti spektro $Y_j^{(i)}$ koeficientai išrenkami iš vaizdo $X$ diskretaus DLGT spektro $Y$

$$Y_j^{(i)} = \left(s_j^{(i)}\, d_j^{(i)}\, d_{2j}^{(i-1)}\, d_{2j+1}^{(i-1)}\, d_{4j}^{(i-2)} \ldots d_{2^{i-1}j}^{(1)}\, d_{2^{i-1}j+1}^{(1)} \ldots d_{2^{i-1}(j+1)-1}^{(1)}\right)^T. \tag{2.2}$$

## 2.2 DLGT SPEKTRO DVIMAČIO VAIZDO FRAGMENTAMS NUSTATYMAS

Vienas iš pagrindinių šio darbo tikslų buvo, DLGT spektro dvimačio vaizdo fragmentams algoritmų sudarymas.

Kaip žinome, vaizdo DLGT spektras pasižymi įdomia savybe. Kiekvienas spektro koeficientas atitinka tam tikra vaizdo fragmentą. Šį ryšį galima užrašyti formaliai:

$$\left[X(m_1,m_2)\right]\ \left(m_1,m_2 \in \{0,1,\ldots,N-1\}, N = 2^n, n \in \square\right) \tag{2.3}$$

yra dvimatis skaitmeninis vaizdas;

$$\left[Y(k_1,k_2)\right]\ \left(k_1,k_2 \in \{0,1,\ldots,N-1\}\right) \tag{2.4}$$

yra dvimatis diskretusis DLGT (su daline dekoreliacija) vaizdo $\left[X(m_1,m_2)\right]$ spektras. Kai $k_r \neq 0\ \left(r \in \{1,2\}\right)$, galima užrašyti – $k_r = 2^{n-i_r} + j_r$, $i_r = \{1,2,\ldots,n\}$, $j_r = \{0,1,\ldots,2^{n-i_r}-1\}$;

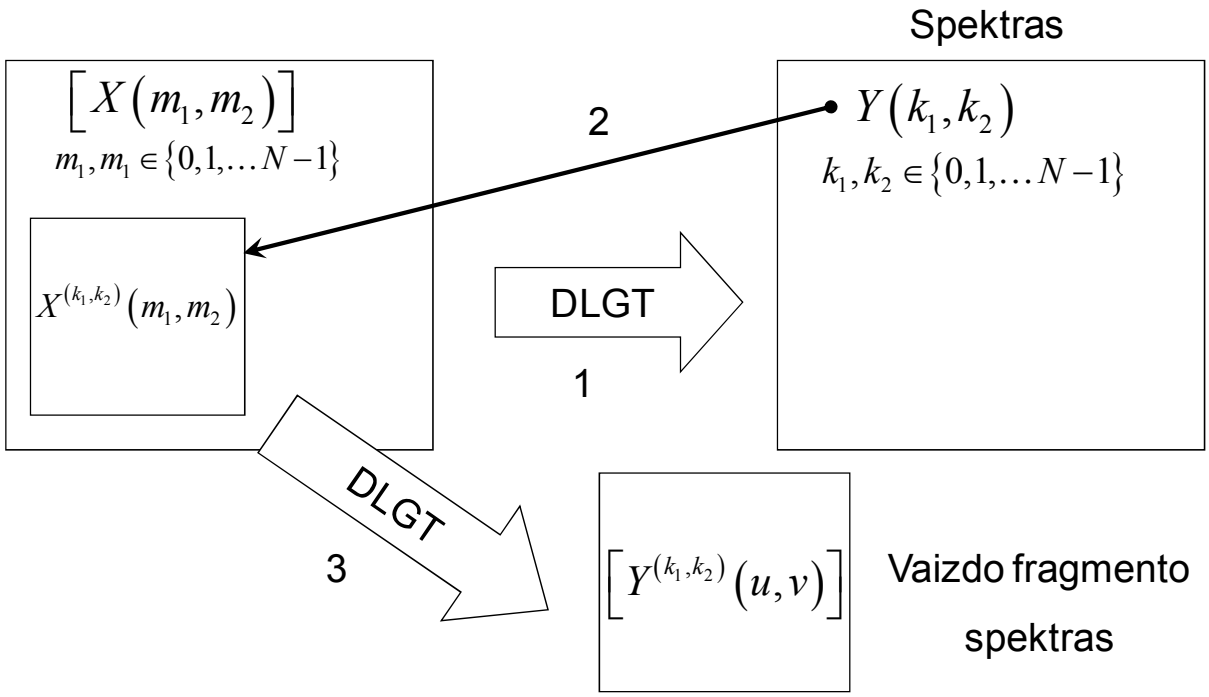
$$\left[X^{(k_1,k_2)}(m_1,m_2)\right]\ \left(\begin{array}{l}(m_1,m_2) \in V_{k_1} \times V_{k_2};\\ \text{čia } V_{k_r} = \{j_r \cdot 2^{i_r}, j_r \cdot 2^{i_r}+1,\ldots,(j_r+1)\cdot 2^{i_r}-1\},\ r = 1,2\end{array}\right) \tag{2.5}$$

yra vaizdo $\left[X\left(m_1,m_2\right)\right]$ fragmentas $2^{i_1}\times 2^{i_2}$, susijęs su spektriniu DLGT koeficientu $Y\left(k_1,k_2\right)$. Reikėtų atkreipti dėmesį, kad $i_1 \geq p_1$, $i_2 \geq p_2$, kai $2^{p_1}\times 2^{p_2}$ $(1 \leq p_1 < n,\ 1 \leq p_2 < n)$ yra minimalus vaizdo $\left[X\left(m_1,m_2\right)\right]$ fragmento, kuriam (skaičiuojant DLGT) netaikoma dekoreliacija, dydis;

$$\left[Y^{(k_1,k_2)}\left(u,v\right)\right]\ \left(u \in \left\{0,1,\ldots,2^{i_1}-1\right\},\ v=\left\{0,1,\ldots,2^{i_2}-1\right\}\right) \tag{2.6}$$

diskretusis vaizdo fragmento $\left[X^{(k_1,k_2)}\left(m_1,m_2\right)\right]$ DLGT spektras.

2.1 paveiksle parodomas vaizdo fragmento spektro radimas, naudojant DLGT tiesiogiai vaizdo fragmentui.

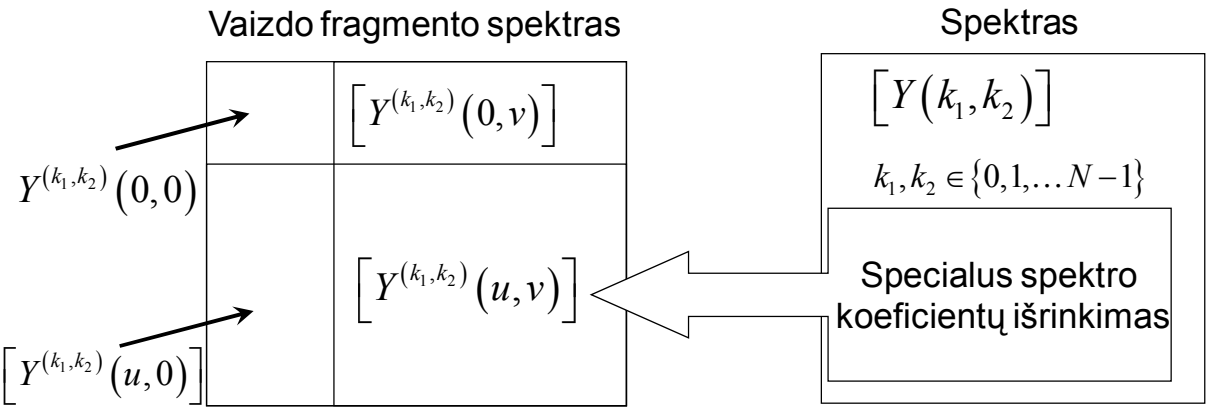



**2.1 pav. Vaizdo fragmento spektro radimo schema**

Pirmiausia taikoma DLGT pradiniam vaizdui $\left[X\left(m_1,m_2\right)\right]$ ir gaunamas spektras $\left[Y\left(k_1,k_2\right)\right]$. Turint omenyje, kad DLGT pasižymi daline lokalizacija erdvėje, bei DLGT metu buvo taikyta dekoreliacija, tai kiekvienas spektro koeficientas $Y\left(k_1,k_2\right)$ atitinka tam tikrą vaizdo fragmentą $\left[X^{(k_1,k_2)}\left(m_1,m_2\right)\right]$, kuris nėra mažesnis nei tai leidžia dekoreliacija. Šioje schemoje 3 žingsniu parodomas pirmas būdas, kaip gauti vaizdo fragmento spektrą, t.y. taikyti DLGT tiesiogiai vaizdo fragmentui.

Kaip buvo minėta, šio darbo tikslas buvo išvesti algoritmą, kurio pagalba žymiai greičiau būtų rastas dvimačio vaizdo fragmento spektras, nenaudojant DLGT tiesiogiai. Šio algoritmo egzistavimą laiduoja transformacijos pilna lokalizacija erdvėje. Kaip žinoma, DLGT pasižymi tik daline lokalizacija. Tačiau panaudojus dekoreliacija, galima sakyti, kad tam tikri spektro koeficientai ir juos atitinkantys tam tikro dydžio vaizdo fragmentai tenkina pilną lokalizaciją erdvėje.

Tyrinėjant vaizdo fragmentus ir jų spektrus $\left[Y^{(k_1,k_2)}(u,v)\right]$ buvo pastebėta, kad šiuos spektrus galima suskaidyti į 4 dalis. Šios dalys vaizduojamos 1 paveiksle:




**2.2 pav. Vaizdo fragmento spektro radimo greitoji procedūra**

Kaip matome, vaizdo fragmento spektrą sudaro kampinis koeficientas $Y^{(k_1,k_2)}(0,0)$, kraštiniai koeficientai $Y^{(k_1,k_2)}(u,0)$ ir $Y^{(k_1,k_2)}(0,v)$, bei spektro blokas $Y^{(k_1,k_2)}(u,v)$. Būtent dėl šio bloko sąsajų su pradinio vaizdo spektro koeficientais, algoritmas veikia žymiai greičiau nei tiesioginė DLGT. Trumpai tariant, fragmento $\left[X^{(k_1,k_2)}(m_1,m_2)\right]$ spektro blokas $\left[Y^{(k_1,k_2)}(u,v)\right]$ yra išrenkamas iš spektro, o kraštiniai ir kampinis koeficientai paskaičiuojami pagal specialias formules ((2.10), (2.11), (2.12) ir (2.13)).

**Algoritmas**

1. Pirmiausia suformuojamos aibės

$$S_V = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-i_1+1}\},\ \alpha_0 = k_1, \alpha_1 = \left\lfloor \frac{\alpha_0}{2} \right\rfloor, \ldots, \alpha_{n-i_1+1} = \left\lfloor \frac{\alpha_{n-i_1}}{2} \right\rfloor, \tag{2.7}$$

$$S_U = \{\beta_0, \beta_1, \ldots, \beta_{n-i_2+1}\},\ \beta_0 = k_2, \beta_1 = \left\lfloor \frac{\beta_0}{2} \right\rfloor, \ldots, \beta_{n-i_2+1} = \left\lfloor \frac{\beta_{n-i_2}}{2} \right\rfloor, \tag{2.8}$$

$$\Im_{k_r} = \{k_r\} \cup \left\{\cup_{q=1}^{i_r-1} \Im_{k_r}(q)\right\}, \tag{2.9}$$

$$\Im_{k_r}(q) = \left\{2^q \cdot k_r, 2^q \cdot k_r + 1, \ldots, 2^q \cdot (k_r+1) - 1\right\},\ r \in \{1,2\}.$$

2. Apskaičiuojami spektriniai DLGT koeficientai vaizdo fragmentui $\left[X^{(k_1,k_2)}(m_1,m_2)\right]$:

$$Y^{(k_1,k_2)}(0,0) = \sum_{q=1}^{n-i_1+1} \sum_{r=1}^{n-i_2+1} (-1)^{\alpha_{r-1}+\beta_{q-1}} \left(\frac{1}{2}\right)^{1\{\alpha_r>0\}} \left(\frac{1}{2}\right)^{1\{\beta_q>0\}} Y(\alpha_r, \beta_q), \tag{2.10}$$

$$Y^{(k_1,k_2)}(u,0) = Y(k_1^*, 0) - \frac{1}{2} \sum_{q=1}^{n-i_1} (-1)^{\beta_{q-1}} Y(k_1^*, \beta_q) \tag{2.11}$$

su visais $u = \{0,1,\ldots,2^{i_1}-1\}$; $k_1^*$ yra $u$-tasis aibės $\Im_{k_1}$ elementas (elementų numeracija aibėje $\Im_{k_1}$ prasideda nuo vieneto);

$$Y^{(k_1,k_2)}(0,v) = Y(0,k_2^*) - \frac{1}{2}\sum_{r=1}^{n-i_2}(-1)^{\alpha_{r-1}}\, Y(\alpha_r, k_2^*) \tag{2.12}$$

su visais $v = \{0,1,\ldots,2^{i_2}-1\}$; $k_2^*$ yra $v$-tasis aibės $\Im_{k_2}$ elementas (elementų numeracija aibėje $\Im_{k_2}$ prasideda nuo vieneto);

$$Y^{(k_1,k_2)}(u,v) = Y(k_1^*, k_2^*) \tag{2.13}$$

su visais $u = \{0,1,\ldots,2^{i_1}-1\}$ ir $v = \{0,1,\ldots,2^{i_2}-1\}$; $k_1^*$ ir $k_2^*$ yra atitinkamai $u$-tasis ir $v$-tasis aibių $\Im_{k_1}$ ir $\Im_{k_2}$ elementai.

## 2.3 PALYGINAMOSIOS EKSPERIMENTINĖS ANALIZĖS REZULTATAI

Šio darbo tikslas buvo išvesti vaizdo fragmento spektro radimo algoritmus ir palyginti jų veikimo greitį su bazine DLGT.

Pažymėkime $\tau_t$ laiko intervalą milisekundėmis, reikalingą tiesiogiai atlikti DLGT ir gauti spektrą. Taip pat $\tau_g$ pažymėkime laiko intervalą milisekundėmis, reikalingą greitajai procedūrai atlikti. Pasitelkus sukurtą programinę įrangą buvo gauti tokie analizės rezultatai:

**2.1 lentelė**

**Greičio testo rezultatai**

| N × N | p = 6 | | p = 7 | | p = 8 | | p = 9 | |
|---|---|---|---|---|---|---|---|---|
| | $\tau_t$ (ms) | $\tau_g$ (ms) | $\tau_t$ (ms) | $\tau_g$ (ms) | $\tau_t$ (ms) | $\tau_g$ (ms) | $\tau_t$ (ms) | $\tau_g$ (ms) |
| 128 × 128 | 25,4728 | 0,4229 | - | - | - | - | - | - |
| 256 × 256 | 25,4797 | 0,4250 | 107,4778 | 1,3556 | - | - | - | - |
| 512 × 512 | 28,4143 | 0,4069 | 101,6958 | 1,2283 | 412,4376 | 5,3197 | - | - |
| 1024 × 1024 | 27,0421 | 0,4954 | 100,4952 | 1,2473 | 421,7992 | 3,9369 | 1662,0160 | 16,9221 |

Kaip matome, gauti rezultatai rodo, kad sukurta greitoji procedūra daug greičiau randa fragmento DLGT spektrą. Pažymėkime $\rho = \frac{\tau_t}{\tau_g}$ algoritmo greičio išlošį ir atvaizduokime duomenis lentele.

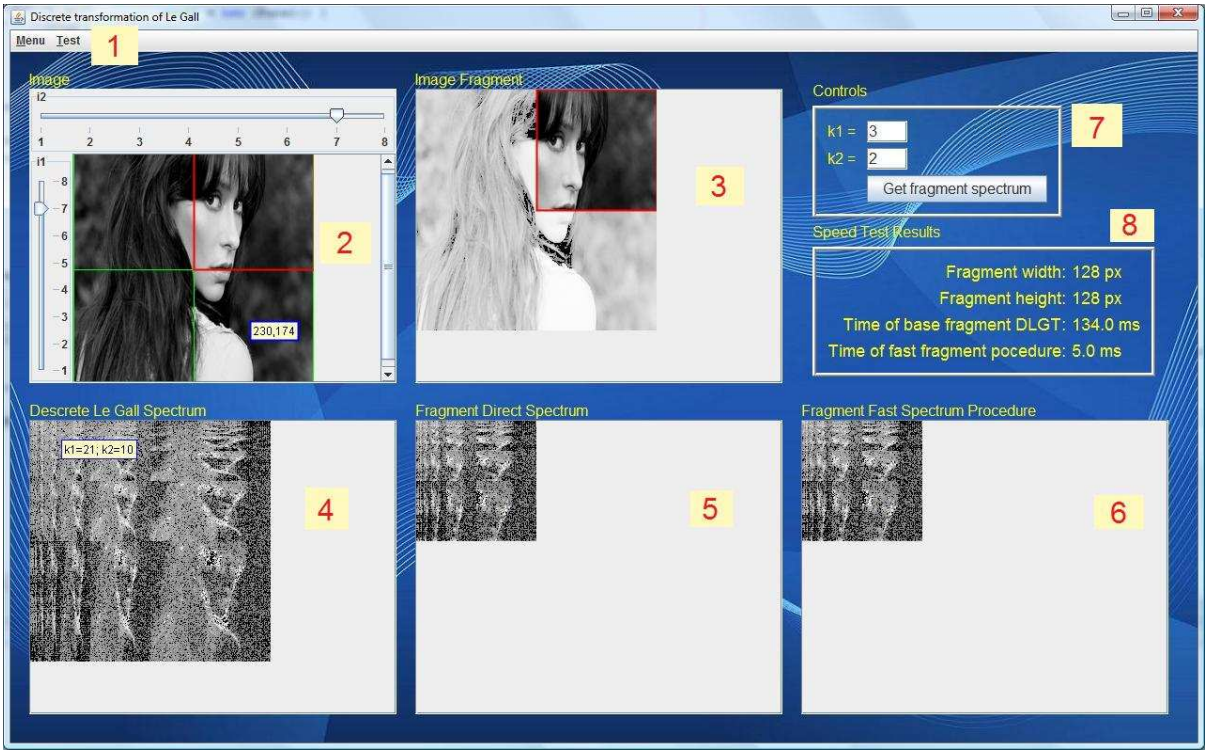
**2.2 lentelė**

**Greičio išlošis**

| N × N | p = 6 | p = 7 | p = 8 | p = 9 |
|---|---|---|---|---|
| | $\rho$ | | | |
| 128 × 128 | 60,24 | - | - | - |
| 256 × 256 | 59,95 | 79,28 | - | - |
| 512 × 512 | 69,83 | 82,79 | 77,53 | - |
| 1024 × 1024 | 54,59 | 80,57 | 107,14 | 98,22 |

Greičio išlošis lyginant algoritmus išties įspūdingas. Be to, nesunkiai galime pastebėti dėsningumą. Kuo norima didesnio fragmento, lyginant su pradiniu vaizdu, spektrą gauti, tuo greičio išlošis didesnis. Tai paaiškinama tuo, kad greitosios procedūros metu, didelė dalis spektro koeficientų tiesiog išrenkama iš pradinio vaizdo spektro, kai tuo tarpu tiesioginė DLGT turi atlikti masyvius skaičiavimus.

# 3 PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI




**3.1 pav. Pagrindinis programos langas**

Pagrindinio programos lango sudedamosios dalys (žr. 3.1 pav.):

1. Meniu. Turi 4 funkcijas: įkelti naują vaizdą, baigti programos darbą, atlikti greičio testą ir atlikti išplėstą greičio testą.
2. Pagrindinio vaizdo laukas. Šiame lauke yra galimybė pažymėti vaizdo fragmentą, kurį norima tirti. Fragmentas žymimas pelės pagalba, pasirinkus norimo dydžio tinklelį. Tinklelis parenkamas naudojant *i1* ir *i2* slinktukais.
3. Pažymėtas vaizdo fragmentas.
4. Diskrečiojo Le Gall spektro laukas. Šiame lauke vaizduojamas spektras, gautas pritaikius Le Gall transformaciją pagrindiniam vaizdui. Be to, lauke veikia interaktyvus spektro koeficiento išrinkimas, t.y. susiejimas su vaizdo fragmentu.
5. Vaizdo fragmento spektro laukas. Šis spektras gautas, pritaikius Le Gall transformaciją vaizdo fragmentui.

6. Vaizdo fragmento spektro laukas. Šis spektras gautas, pritaikius greitąją procedūrą.

7. Fragmento valdymo laukas. Galima įrašyti spektro koeficientus, pagal kuriuos išrenkamas vaizdo fragmentas.

8. Greičio testo rezultatų laukas. Atlikus greičio testą pasirinktam fragmentui, parodomi rezultatai.

**Darbas su programa**

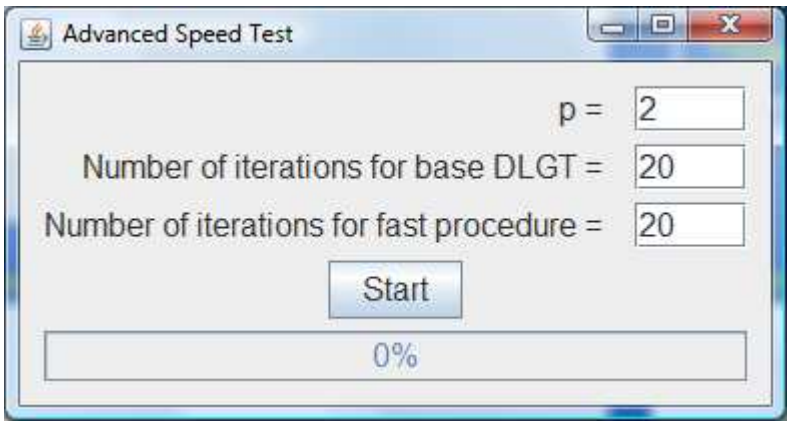
Darbas pradedamas paleidus *LeGallWavelets.jar* vykdomąjį failą. Atsidariusiame lange automatiškai paruošiamas pilkų tonų 256×256 dydžio vaizdas analizei. Bet kuriuo programos naudojimo metu galima išsirinkti pageidaujamą vaizdą, pasirinkus *Meniu* punktą − *Open Image*. Vaizdo analizei tinka šie 4 skaitmeninių vaizdų formatai: *jpeg*, *bmp*, *gif* ir *png*. Pasirinkus naują vaizdą, tenka luktelėti (priklausomai nuo vaizdo dydžio), kol programa suskaičiuoja vaizdo DLGT spektrą, ir jį atvaizduoja spektro lauke *Discrete Le Gall Spectrum*. Pasibaigus spektro skaičiavimams, vartotojas turi 3 galimybes pasirinkti vaizdo fragmentą:

1. tiesiogiai iš vaizdo, pasirinkus norimo dydžio tinklelį slinktukų pagalba;
2. *Controls* lauke įvedus pageidaujamus spektro koeficientus *k1* ir *k2*;
3. spustelėjus norimą spektro koeficientą iš spektro lauko.

Kai tik vartotojas išsirenka vaizdo fragmentą, programa iš karto suskaičiuoja ir atvaizduoja vaizdo fragmento tiesioginį DLGT spektrą ir gautą, panaudojus greitąja skaičiavimo procedūrą. Atitinkamuose laukuose vartotojas gali pamatyti ir palyginti suskaičiuotus spektrus. Tam tikslui sukurtas informacinis laukelis, kuriame rodoma spektro koeficiento reikšmė esanti po pelės žymekliu.

Programinė įranga skirta palyginamajai analizei atlikti, t.y. greičio testams atlikti. Programoje yra įgyvendinti 2 greičio testai: paprastasis ir sudėtingasis. Prieš paprastojo testo vykdymą, vartotojas turi buti išsirinkęs vaizdo fragmentą, kurio pagalba bus lyginami tiesioginės DLGT ir greitosios procedūros atlikimo laikai. Šio testo metu, programa automatiškai pasirenka iteracijų skaičių, kuris priklauso nuo analizuojamo vaizdo ir jo fragmento dydžio. Gauti laikai atvaizduojami *Results* lauke.

Sudėtingojo greičio testo darbas pradedamas nurodžius keletą parametrų: mažiausią dekoreliuojamo vaizdo fragmento dydį $p$, bei iteracijų skaičius tiesioginei DLGT ir greitajai procedūrai. Šio testo rezultatai spausdinami į du *txt* formato bylas, kuriuose nurodomas vaizdo dydis, visų galimų fragmentų dydžiai, koeficientai *k1* ir *k2*, bei laikas milisekundėmis.



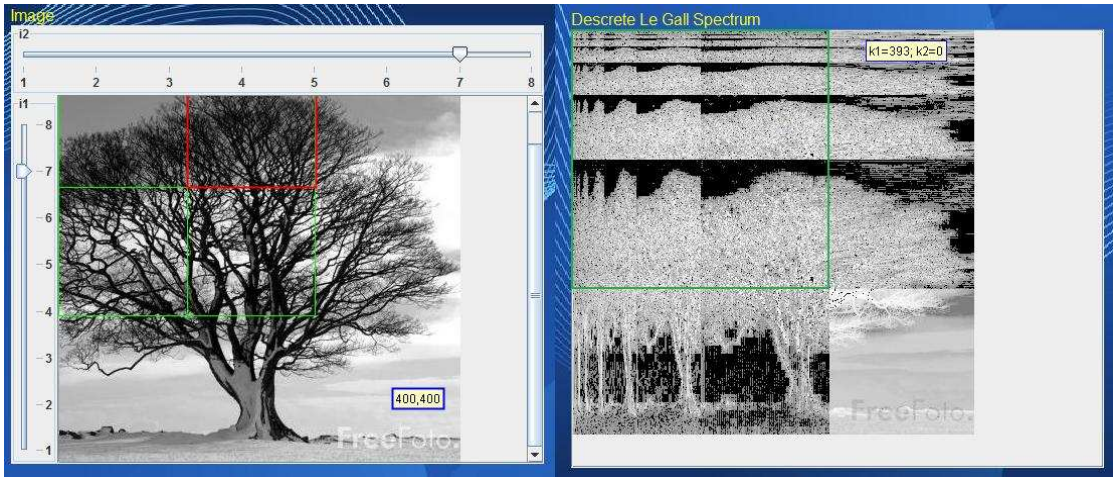

**3.2 pav. Sudėtingojo greičio testo valdymo langas**

**Pastabos**

Ši programinė įranga buvo sukurta naudojantis Java programavimo kalbą. Viena iš pagrindinių klasių darbui su vaizdais yra *BufferedImage*. Naudojantis šia klase, galima nesunkiai redaguoti skaitmeninius vaizdus. Be to, skaitmeninį vaizdą galima paversti į sveikųjų skaičių matricas.

Viena iš populiariausių vaizdų kodavimo sistemų yra *RGB*, todėl naudojantis tokia sistema, gaunama 3 vaizdų matricos, t.y. po vieną matricą raudonai, žaliai ir mėlynai spalvoms. Tokiu būdu, norint gauti DLGT spektrą spalvotam vaizdui, tenka atlikti transformaciją 3 kartus, kiekvienos spalvos matricai. Buvo pastebėta, kad *bmp* ir *jpg* formato pilkų tonų vaizdai pasižymi savybe, kad visos spalvų 3 matricos yra vienodos. Kuriant programinę įrangą buvo į tai atsižvelgta: spalvotiems vaizdams DLGT taikoma 3 kartus, o nespalvotiems − 1 kartą, o gautas spektras, pareplikuojamas dvejoms likusioms spalvoms.

Teoriškai DLGT transformacija skirta tik $N = 2^n$ ($n \in \square$) vaizdams. Tačiau dėl didesnio programinės įrangos universalumo, galima analizuoti bet kokio dydžio vaizdus. Programa automatiškai „apkerpa“ netinkamo dydžio vaizdus, ir išrenka maksimalų tinkamą dydį transformacijai. 3.3 paveiksle analizuojamas 400*400 pikselių vaizdas. Kaip matome, programa žalia spalva pažymėjo analizei tinkamą vaizdo plotą (256*256 pikselių).




**3.3 pav. 400*400 pikseliu vaizdo ir spektro pavyzdys.**

# IŠVADOS

1. Visos diskrečiosios bangelių transformacijos (DBT) pasižymi (pilno arba dalinio) lokalizavimo erdvėje savybėmis. Pilnas lokalizavimas erdvėje būdingas pačiai paprasčiausiai DBT − Haaro transformacijai. Aukštesniųjų eilių DBT būdingas dalinis lokalizavimas erdvėje. Kaip ir kitos DBT, diskrečioji Le Gall transformacija pasižymėjo daliniu lokalizavimu erdvėje, todėl šiai problemai spręsti buvo pasitelkta dekoreliacija.

2. Remiantis pagerintomis diskrečiosios Le Gall transformacijos lokalizavimo savybėmis buvo sudaryta greitoji DLGT procedūra vaizdo fragmento spektrui gauti. Eksperimento metu įrodyta, kad sudaryta procedūra veikia daugiau nei 50 kartų greičiau nei tiesioginis DLGT taikymas vaizdo fragmentui.

3. Sudaryta ir realizuota greitoji procedūra atveria naują praktinio taikymo sritį − lokaliai progresyvaus vaizdų kodavimą. Viena iš šio kodavimo pritaikymo krypčių būtų mažo pralaidumo kompiuteriniuose kanaluose, kuriuose vaizdų glaudinimas yra ypač svarbus.

# LITERATŪRA


1. Diskrečiosios transformacijos: mokomoji knyga / Jonas Valantinas - Kaunas : Kauno technologijos universitetas, 2008. – 86 p.
2. Jonas Valantinas, On the Use of Le Gall Wavelets in Implementing Locally Progressive Digital Signal Coding Idea. Image and Signal Processing and Analysis, 2009. ISPA 2009. Proceedings of 6th International Symposium on. Sazburg, 2009 – 12-16 p.
3. Daubechies, I., Ten Lectures on Wavelets, SIAM, Philadelphia, 1992.
4. C. Valens, 1999-2010, A Really Friendly Guide to Wavelets. [Žiūrėta 2011 05 15]. Prieiga per internetą: http://polyvalens.pagesperso-orange.fr/clemens/wavelets/wavelets.html
5. Chan, Y. T., Wavelet Basics, Kluwer Academic Publisher Norwell, Ma, 1995.
6. Burrus, C. S. and R. A. Gopinath, H. Guo, Introduction to wavelets and wavelet transforms, A primer, Upper Saddle River, NJ(USA): Prentice Hall, 1998

# PRIEDAI

Programinės įrangos kodas skirtas Java programavimo kalbai.

Algoritmas skirtas DLGT transformacijai įvykdyti.

```
package DLGT;
public class LeGall53MatrixWithm {
    int m;
    int N, M;
    double XX[][][]; // image matrix
    double YY[][][]; // spectrum matrix

    public LeGall53MatrixWithm(double XX[][][], int m) {
        this.XX = XX;
        this.N = XX[0].length;
        this.M = XX[0][0].length;
        this.m = m;
        this.YY = new double[3][N][M];
    }

    public void makeDLGT() {
        if ((XX[0][0][0] == XX[1][0][0]) && (XX[0][0][0] == XX[2][0][0])) {
            if ((N > 1) && (M > 1)) {
                double X[] = new double[N];
                // DLGT by columns
                for (int j = 0; j < M; j++) {
                    for (int i = 0; i < N; i++) {
                        X[i] = XX[0][i][j];
                    }
                    X = DLGTarray(X, N, m);
                    for (int i = 0; i < N; i++) {
                        XX[0][i][j] = X[i];
                    }
                }
                // DLGT by rows
                X = new double[M];
                for (int i = 0; i < N; i++) {
                    X = DLGTarray(XX[0][i], M, m);
                    XX[0][i] = X;
                }
                //double YY[][]=new double[N][N];
                XX[1] = XX[0];
                XX[2] = XX[0];
                YY = XX;
            }
        } else {
            if ((N > 1) && (M > 1)) {
                for (int k = 0; k < 3; k++) {
                    double X[] = new double[N];
                    // DLGT by columns
                    for (int j = 0; j < M; j++) {
                        for (int i = 0; i < N; i++) {
                            X[i] = XX[k][i][j];
                        }
                        DLGTarray(X, N, m);
                        for (int i = 0; i < N; i++) {
                            XX[k][i][j] = X[i];
                        }
                    }
                    // DLGT by rows
```

```
                    X = new double[M];
                    for (int i = 0; i < N; i++) {
                        X = DLGTarray(XX[k][i], M, m);
                        XX[k][i] = X;
                    }
                    YY[k] = XX[k];
                }
            }
        }
    }

    public double[] DLGTarray(double X[], int N, int m) {
        int n = (int) (Math.log(N) / Math.log(2));
        double O[][] = new double[(int) Math.pow(2, n - 1)][n];
        double E[][] = new double[(int) Math.pow(2, n - 1)][n];
        double D[] = new double[(int) Math.pow(2, n - 1)];
        double S[][] = new double[(int) Math.pow(2, n - 1)][n + 1];
        for (int i = 0; i < (int) Math.pow(2, n - 1); i++) {
            O[i][0] = X[2 * i];
            E[i][0] = X[2 * i + 1];
        }
        int alpha[] = new int[n + 1];
        int beta[] = new int[n + 1];
        for (int i = 1; i <= n; i++) {
            if (i <= m) {
                alpha[i] = (int) Math.pow(2, m - i);
                beta[i] = (int) Math.pow(2, n - m);
            } else {
                alpha[i] = 1;
                beta[i] = (int) Math.pow(2, n - i);
            }
            int temp1[] = new int[beta[i] + 2];
            int temp2[] = new int[beta[i] + 2];
            for (int t = 1; t <= beta[i]; t++) {
                temp1[t] = alpha[i] * t - 1;
                temp2[t] = alpha[i] * (t - 1);
            }
            int t = 1;
            for (int k = 0; k < (int) Math.pow(2, n - i); k++) {
                if (k == temp1[t]) {
                    D[k] = E[k][i - 1] - O[k][i - 1];
                    t++;
                } else {
                    D[k] = E[k][i - 1] - (O[k][i - 1] + O[k + 1][i - 1]) / 2;
                }
            }
            t = 1;
            for (int k = 0; k < (int) Math.pow(2, n - i); k++) {
                if (k == temp2[t]) {
                    S[k][i] = O[k][i - 1] + D[k] / 2;
                    t++;
                } else {
                    S[k][i] = O[k][i - 1] + (D[k - 1] + D[k]) / 4;
                }
            }
            for (int k = 0; k < (int) Math.pow(2, n - i - 1); k++) {
                O[k][i] = S[2 * k][i];
                E[k][i] = S[2 * k + 1][i];
            }
            System.arraycopy(D, 0, X, (int) Math.pow(2, n - i), (int) Math.pow(2,
n - i));
        }
        //Creating spectrum
        X[0] = S[0][n];
```

```
        return X;
    }

    public double[][][] getDLGTSpectrum() {
        return YY;
    }
}
```

Vaizdo fragmento DLGT spektro greitasis algoritmas.

```
package DLGT;
import java.awt.Point;
public class FragmentFastDLGT {
    double Y[][][];
    double Yk1k2[][][];
    int k[] = new int[2];
    int N;
    int n;

    public FragmentFastDLGT(double[][][] spectrumMatrix, Point k1k2) {
       this.Y = spectrumMatrix;
       this.N = spectrumMatrix[0].length;
       this.n = (int) (Math.log(this.N) / Math.log(2));
       this.k[0] = k1k2.x;
       this.k[1] = k1k2.y;
       this.Yk1k2 = new double[3][N][N];
    }

    public void makeFragmentFastDLGT() {
        if ((Y[0][0][0] == Y[1][0][0]) && (Y[0][0][0] == Y[2][0][0])) {
            Yk1k2[0] = FastDLGT(n, k, Y[0]);
            Yk1k2[1] = Yk1k2[0];
            Yk1k2[2] = Yk1k2[0];
        } else {
            for(int q = 0; q < 3; q++) {
                Yk1k2[q] = FastDLGT(n, k, Y[q]);
            }
        }

    }
    public double[][][] getFragmentFastDLGTSpectrum() {
        return Yk1k2;
    }
    public double[][] FastDLGT(int n, int k[], double Y[][]) {
        int i[] = {n, n};
        int j[]= {0, 0};
```

```
        for(int p = 0; p < 2; p++) {
            for(int t = n; t >= 0; t--) {
                if(Math.pow(2,n-t)>k[p]) {
                    i[p]=t+1;
                    break;
                }
            }
            j[p]=k[p]-(int)(Math.pow(2,n-i[p]));
        }
        int Yk1[] = new int[(int)Math.pow(2,i[0])-1];
        int Yk2[] = new int[(int)Math.pow(2,i[1])-1];
        int alfa[] = new int[n+2-i[0]];
        int beta[] = new int[n+2-i[1]];

        int t = 0;
        for(int p = 0; p < i[0]; p++) {
            for(int q = k[0]*(int)(Math.pow(2,p)); q <
(k[0]+1)*(int)(Math.pow(2,p)); q++) {
                Yk1[t] = q;
                t++;
            }
        }
        t = 0;
        for(int p = 0; p < i[1]; p++) {
            for(int q = k[1]*(int)(Math.pow(2,p)); q <
(k[1]+1)*(int)(Math.pow(2,p)); q++) {
                Yk2[t] = q;
                t++;
            }
        }
        alfa[0] = k[0];
        for(int p = 1; p < n+2-i[0]; p++) {
            alfa[p] = (int)Math.floor(alfa[p-1]/2);
        }
        beta[0] = k[1];
        for(int p = 1; p < n+2-i[1]; p++) {
            beta[p] = (int)Math.floor(beta[p-1]/2);
        }
        double Yk1k2[][] = new double[Yk1.length+1][Yk2.length+1];
        for(int p = 0; p < Yk1.length+1; p++) {
            for(int q = 0; q < Yk2.length+1; q++) {
                if((p == 0) && (q == 0)) {
                    double sum = 0;
```

```
                    for(int r1 = 1; r1 < n-i[0]+2; r1++) {
                        for(int r2 = 1; r2 < n-i[1]+2; r2++) {
                            sum += Math.pow(-1,alfa[r1-1])*Math.pow(-1,beta[r2-
1])*

Math.pow(0.5,Math.ceil(alfa[r1]/(alfa[r1]+0.1)))*Math.pow(0.5,Math.ceil(beta[r2]/(
beta[r2]+0.1)))*
                                    Y[alfa[r1]][beta[r2]];
                        }
                    }
                    Yk1k2[p][q] = sum;
                } else if(p == 0) {
                    double sum = 0;
                    for(int r = 1; r < n-i[0]+1; r++) {
                        sum += -0.5*Math.pow(-1,alfa[r-1])*Y[alfa[r]][Yk2[q-1]];
                    }
                    Yk1k2[p][q] = Y[0][Yk2[q-1]] + sum;
                } else if(q == 0) {
                    double sum = 0;
                    for(int r = 1; r < n-i[1]+1; r++) {
                        sum += -0.5*Math.pow(-1,beta[r-1])*Y[Yk1[p-1]][beta[r]];
                    }
                    Yk1k2[p][q] = Y[Yk1[p-1]][0] + sum;
                } else {
                    Yk1k2[p][q]=Y[Yk1[p-1]][Yk2[q-1]];

                }
            }
        }
        return Yk1k2;
    }
}
```